// DESCARGA GRATUITA //

"int año = 1, número 1;" /* ENERO 2008 */

µCONTROL

Electrónica en General Pics en Particular

Construye tus propios circuitos

Ideal para aficionados que desean fabricar sus propios PCB

Usando LCDs

Características Principales, usos y aplicaciones.

PIC BASIC

Completo tutorial para programar microcontroladores en BASIC

.indice

.staff

hicimos este número...

Dirección y Redacción:
Ariel Palazzesi
Argentina
arielpalazzesi@gmail.com
www.ucontro.com.ar

Redactores:
Reiner Torres Labrada
Cuba
reinertl@gmail.com

Mario Sacco
Argentina
service.servisystem@gmail.com

Carlos Ortega Sabio
España
carlos.ortegasabio@ucontrol.com.ar

Diego Márquez García - Cuervo
picmania@garcia-cuervo.com
www.picmania.garcia-cuervo.net

Marcos Lazcano
Argentina
marcos.lazcano@gmail.com

Pedro Palitroquez
Venezuela
palitroquez@gmail.com

Diseño:
Verónica C. Lavore
Argentina
azimut.estudio@gmail.com

.editorial

Por fin, y luego de varios meses de trabajo, el primer número de la Revista uControl está en tus manos. Todo proyecto nuevo necesita de un tiempo de maduración, y este no ha sido la excepción.

Si bien desde el principio el concepto de la revista estaba claro, faltaba pulir decenas de detalles. Sabíamos que íbamos a escribir una revista que le fuese de utilidad al estudiante, al hobbista y también al que ya sabe bastante de electrónica. Teníamos como premisa utilizar un lenguaje claro, ameno, que permitiese establecer una comunicación efectiva entre quien escribe y quien lee. Íbamos a publicar mucha información, que si bien a veces está disponible en la web, no se encuentra en español o bien no está completa.

Sin embargo, una cosa es saber que se quiere hacer, y otra muy diferente es transformar esa idea en algo concreto. Aunque suene extraño, lo más fácil fue escribir todo el contenido que llena las páginas de este ejemplar. La tarea de diseño grafico (¡gracias Verónica!) fue todo un desafío. De nada servía tener buenos artículos si resultaban difíciles de leer, culpa de una tipografía desafortunada o por una maquetación equivocada.

Todos los que formamos parte del staff de esta revista nos dedicamos a la electrónica. Algunos programan microcontroladores, otros desarrollan proyectos para empresas, y los más afortunados incluso obtienen sus ingresos de esta fascinante ciencia que es la electrónica. Pero es la primera vez que trabajamos juntos. Y para muchos, también es la primera vez que publican un trabajo en una revista.

Estamos orgullosos del resultado de nuestro trabajo, y sabemos que a medida que transitemos este camino que hoy inauguramos, el resultado será aún mejor.
Este es el primer ejemplar de una publicación que esperamos este mucho tiempo entre nosotros. Aportando un pequeño granito de arena en tu trabajo, en tu carrera o en tu hobby. Está claro que se trata de un objetivo ambicioso, y que no podremos lograrlo sin tu ayuda. Para nosotros es muy importante conocer tu opinión y saber cuáles son los temas que te gustaría ver desarrollados en los números siguientes. Puedes enviarnos tus sugerencias, críticas o colaboraciones a *revista@ucontrol.com.ar*. Prometemos contestar todos los mails.

El contenido de las notas publicadas, salvo indicación contraria, es de libre distribución. Eso significa que puedes usarlo para lo que quieras, aunque eso sí, citando a su autor y a la revista. A lo largo del 2008 cada 64 días exactos habrá un nuevo ejemplar de la Revista uControl. Ese es nuestro compromiso.
¡Hasta el próximo número!

"el relojito" primera parte

"El relojito" es un reloj de pared bastante especial. Además de dar la hora (como todo reloj que se precie de tal), también nos muestra la temperatura ambiente. Pero lo que lo hace diferente a la mayoría de los relojes electrónicos, cuyos esquemas puedes encontrar navegando por la web, es la forma en que está construido su particular segundero. En efecto, en lugar de indicar el trans-curso de los segundos mediante un par de display LED de 7 segmentos, como es habitual, lo hace mediante 60 diodos LED dispuestos en forma de circulo a lo largo del borde exterior del circuito impreso que aloja todos los componentes del reloj.

.Descripción del proyecto

El reloj que vamos a construir puede indicar la hora y los minutos mediante 4 display LED de 7 segmentos, en el formato "HH:MM", donde los ":" centrales están constituidos por dos pequeños LEDs de 3mm. Estos displays también se utilizan para mostrar la temperatura, que se obtiene mediante un sensor de temperatura Dallas DS1820. Para mantener funcionando el reloj con una exactitud razonable se ha utilizado un reloj de tiempo real DS1307.

El segundero, como decíamos, es una circunferencia formada por 60 LEDs de 5mm, controlados solamente con dos pines del PIC16F628A. Esto es posible gracias a que se utiliza de un registro de desplazamiento construido con 8 circuitos integrados 74HC164N. En este mismo número de uControl encontraras la explicación de su funcionamiento.

"El relojito", listo para comenzar a funciona

Además, hemos dotado al reloj de 4 pequeños pulsadores, que servirán para llevar a cabo las tareas de puesta en hora, selección del modo de funcionamiento, etc.

Hemos dividido este artículo en dos partes, ya que de no hacerlo resultaría muy extenso. En este número explicaremos detalladamente el funcionamiento del hardware, y en el número siguiente veremos paso a paso como programarlo.

.El circuito

Si bien el circuito del Relojito puede resultar intimidante por su tamaño (emplea 18 circuitos integrados y más de 100 resistores) en realidad no es tan complejo como parece.

El corazón del esquema es un PIC16F628A, que se encarga de llevar a cabo todas las tareas necesarias.

Este microcontrolador emplea como oscilador generador de pulsos de reloj un cristal de 4MHz, con dos condensadores de 22pF, conectados a los pines 15 y 16.

Los pulsadores encargados de la gestión de la puesta en hora y selección del modo de funcionamiento se encuentran conectados a los pines 10, 11, 12 y 13, que corresponden a los bits 4, 5 ,6 y 7 del PORTB. Cada uno de estos pines se ha puesto a GND mediante un resistor de 10K, al presionar un pulsador, el pin correspondiente se pone a +V.

La temperatura se lee desde un sensor Dallas DS1820, conectado al pin 4 del microcontrolador. Este pin corresponder al bit 5 del PORTA.

Este es el esquema eléctrico de nuestro reloj *(puedes descargarlo desde www.ucontrol.com.ar)*

Para mantener la exactitud del reloj empleamos un pequeño circuito integrado, también de Dallas, que se encarga de contar el tiempo por nosotros. Se trata del DS1307, de 8 pines, que dispone de su propio cristal (de 32.768 KHz) y de una pila CR-2032 de 3V de respaldo. Esta pila proporciona la energía necesaria para que el DS1307 siga funcionando en caso de falta de energía proveniente de la fuente de alimentación principal. Esto evitará tener que volver a poner en hora el reloj cada vez que lo desenchufemos de la red eléctrica.

Para mostrar tanto la información correspondiente a las horas y minutos como los datos de la temperatura, se emplearon 4 display LED de 7 segmentos de unos 3.5 centímetros de altura. El modelo elegido fue el C-1021H de Paralight. Se trata de display de cátodo común, donde cada segmento está constituido por dos LEDs rojos en serie. Los ":" centrales están formados por dos LEDs de 3mm conectados en serie, del mismo color que los displays, y son manejados desde el pin 7 del microcontrolador (PORTB.1)

Dado que el multiplexar estos displays mediante las técnicas tradicionales hubiese exigido un elevado número de pines de E/S del microcontrolador PIC16F628A, se utilizó un registro de desplazamiento construido a partir de cuatro circuitos integrados 74HC164N conectados en cascada. Cada una de las salidas de estos integrados controla uno de los segmentos de los displays. El pin 17 del microcontrolador (bit 0 del PORTA) se encarga de proporcionar los datos al registro de desplazamiento, mientras que el pin 18 (bit 1 del PORTA) entrega los pulsos de reloj necesarios.

El mismo truco del registro de desplazamiento se utilizó para controlar los 60 LEDs que conforman el segundero. Esta vez fueron necesarios 8 circuitos integrados 74HC164N. Este registro dispone de 64 salidas, de las que se aprovechan solo las primeras 60. Cada una de estas salidas controla uno de los LEDs a través de un resistor que limita la corriente que los atraviesa.

Dado que el consumo máximo posible de este proyecto es bastante elevado para lo que estamos acostumbrados, hemos dividido la etapa de alimentación en tres partes, cada una de ellas encargada de proporcionar energía a una de las secciones del reloj.

De esta manera, una de las etapas construidas alrededor de un regulador de voltaje LM7805 proporciona la corriente que necesita el microcontrolador, el sensor de temperatura y el reloj de tiempo real. Otra de las etapas alimenta los displays y el registro de desplazamiento que lo controla, y la tercera hace lo propio con los LEDs del segundero y sus circuitos integrados de control.

Si bien no figuran en el esquema eléctrico, hemos colocado condensadores cerámicos de 0.1uF entre los pines de alimentación de cada uno de los 74HC164N. Se pueden ver en las fotos, están soldados directamente sobre las pistas del PCB.

.El circuito impreso

Como es de suponer, para albergar todo esto hace falta un circuito impreso bastante grande. Dado que en uControl intentamos mantener las cosas simples, por lo general no utilizamos circuitos integrados ni componentes de montaje superficial ni PCB de doble faz. Esto también ayudó a que el PCB sea grande.

Diseño del PCB. Puedes descargarlo de www.ucontrol.com.ar, listo para imprimir.

Quizás el rasgo más representativo de este circuito impreso sea su forma, ya que nos hemos apartado del clásico diseño rectangular o cuadrado y lo hemos dibujado como una circunferencia, de forma que los LEDs del segundero adopten la misma disposición que tendrían las marcas en la esfera de un reloj de aguja.

El diámetro de la placa de circuito impreso es de 18 centímetros. Cortarla con forma de circunferencia es bastante trabajoso, pero no imposible. De todos modos, aquellos que no se animen a cortar el PCB con esta forma, pueden simplemente utilizar un PCB cuadrado de 18 centímetros de lado con el dibujo que proponemos en el centro.

Para construir el PCB basta con utilizar el diseño que puede descargarse en formato PDF desde uControl, e imprimirlo siguiendo los pasos de nuestro tutorial "Como construir tus propios PCB".

.Componentes

La lista de componentes que vamos a emplear es bastante extensa, pero afortunadamente se trata de componentes de bajo costo, por lo que se trata de un proyecto al alcance de todos los bolsillos.

El PCB puede ser construido utilizando el método que explicamos aquí.

Estos son los componentes que utilizaremos.

La lista de materiales necesarios:

12 circuitos integrados 74HC164N.
92 resistores de 220 ohm, 1/8 de Watt.
7 resistores de 10K, 1/8 de Watt.
1 circuito integrado DS1307.
1 microcontrolador PIC16F628A.
1 sensor de temperatura DS1820
1 cristal de 4 MHz.
1 cristal de 32.768 KHz.
14 condensadores cerámicos de 100 nF (0.1uF).
2 condensadores cerámicos de 22pF.
1 condensador electrolítico de 220uF/16V.
1 zócalo para pila CR-2032
4 displays de cátodo común C-1021H de Paralight.
2 LEDs rojos de 3mm.
60 LEDs rojos de 5mm.
1 diodo 1N4001.
1 bornera para circuito impreso de dos tornillos.
4 pulsadores de 8mm para circuito impreso.
3 reguladores de voltaje LM7805.

También necesitaras un trozo de PCB virgen de una sola cara, con un tamaño de 18x18 centímetros, y zócalos para los circuitos integrados

.Montaje

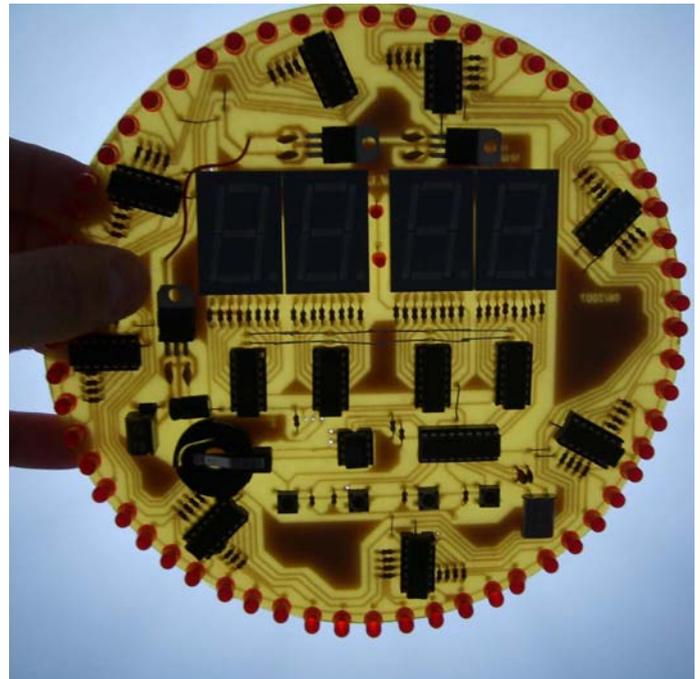
No hay complejidades importantes asociadas al montaje de los componentes de este proyecto. Una buena idea es comenzar el trabajo de soldadura por los puentes y zócalos, para luego seguir con los resistores y condensadores. Los displays, reguladores de voltaje y LEDs deberían montarse en último lugar, cuidando de que estén en la posición correcta. Será un trabajo que tomara al menos una o dos horas, así que hay que encararlo con paciencia.

Una vez que todo esté en su lugar, y antes de colocar los circuitos integrados en sus zócalos, podemos alimentar el relojito y comprobar que a la salida de cada regulador de voltaje tenemos 5V. También podemos medir la tensión en los pines de los zócalos encargados de alimentar a cada integrado, para no tener alguna sorpresa desagradable. Entre los pines 7 y 14 de cada 74HC164N debe haber 5V, lo mismo que entre los pines 5 y 14 del zócalo correspondiente al microcontrolador.

Si todo está bien, podemos poner cada integrado en su sitio, cuidando ponerlos en la dirección correcta.

.Conclusión

Hemos terminado con el montaje del hardware de nuestro relojito. En el próximo número de la revista aprenderemos a programarlo.

Mientras tanto, puedes leer los artículos sobre registros de desplazamiento que publicamos en este ejemplar: seguramente te serán de utilidad para comprender el funcionamiento del programa de este proyecto.

En el próximo número veremos cómo programar el microcontrolador.

sistema de protección de altavoces

Nunca estamos libres de que ocurra algún imprevisto en la salida de audio de nuestra cadena de sonido, deteriorando o destruyendo los parlantes. En éste artículo, proponemos un sistema de protección similar al que traen solo los equipos HI-FI reloj.

Cuando un circuito amplificador de audio, de concepción moderna presenta problemas, es común que alguno de los transistores de la etapa final de salida haya entrado en cortocircuito. Los amplificadores construidos en base a un circuito integrado, también están comprendidos dentro de esta generalidad.

Como resultado del problema planteado, aparecerá a la salida del amplificador una tensión de corriente continua elevada. Si esta tensión no es interrumpida a tiempo, terminará destruyendo las bobinas de los parlantes por exceso de temperatura.

Los sistemas que utilizan los equipos Hi-Fi detectan la presencia de corriente continua en algunas de las salidas de audio.

El circuito propuesto, es una adaptación de los sistemas que en la actualidad utilizan los equipos de audio Hi-Fi, los cuales tienen como misión, detectar la presencia de corriente continua en alguna de las salidas de audio. Si por alguna razón, en la salida de audio del sistema apareciese una tensión de directa, por un tiempo prolongado, el circuito protector se activará y pasará al equipo al modo Stand-By, cortando abruptamente la alimentación del mismo.

El esquema de este diseño puede verse en la **Figura 1**, pero para una mejor comprensión durante la explicación del mismo, utilizaremos el esquema de la **Figura 2.**

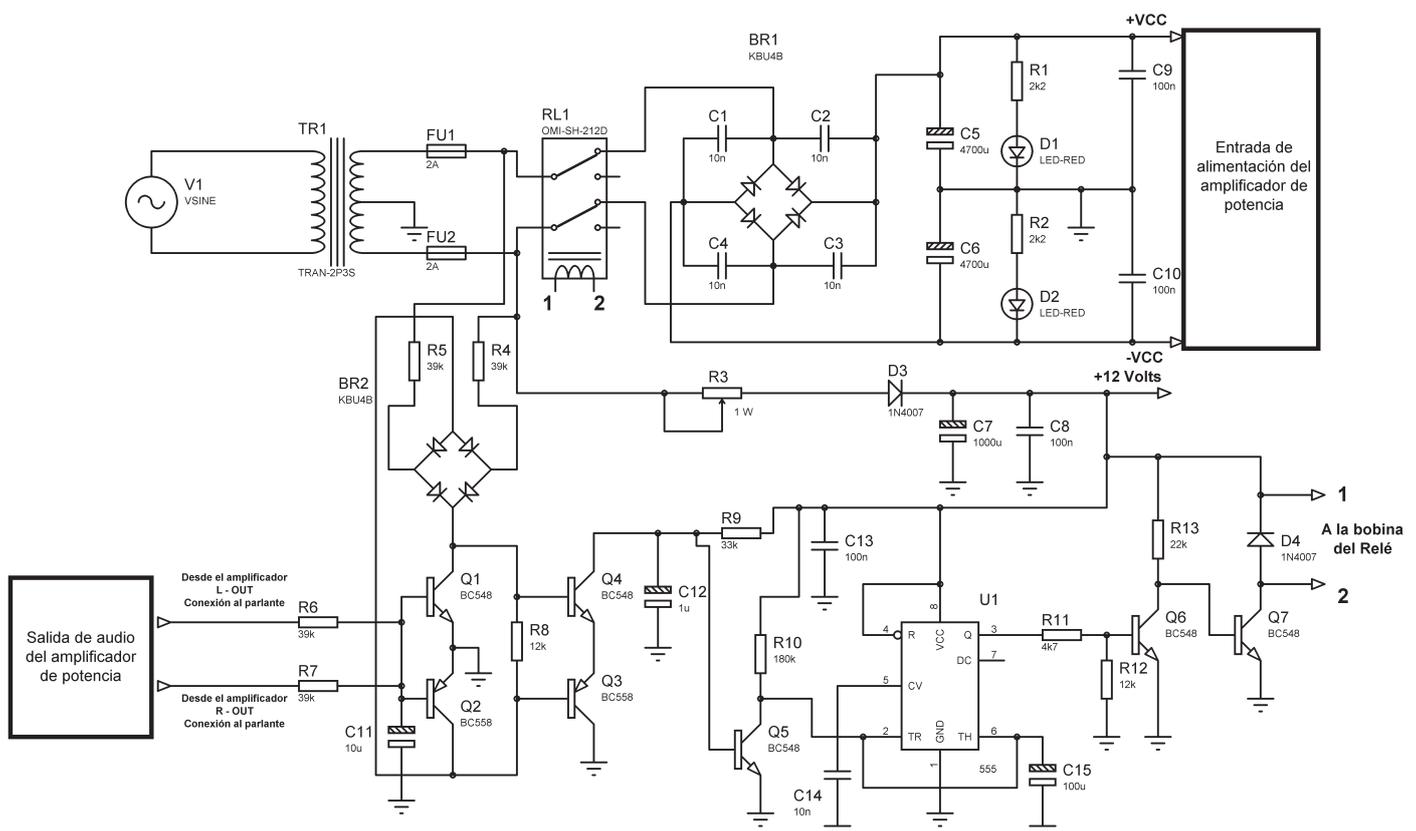



FIGURA 1: Un puñado de transistores bastará para proteger nuestro sistema de altavoces. (Descargalo de www.ucontrol.com.ar)

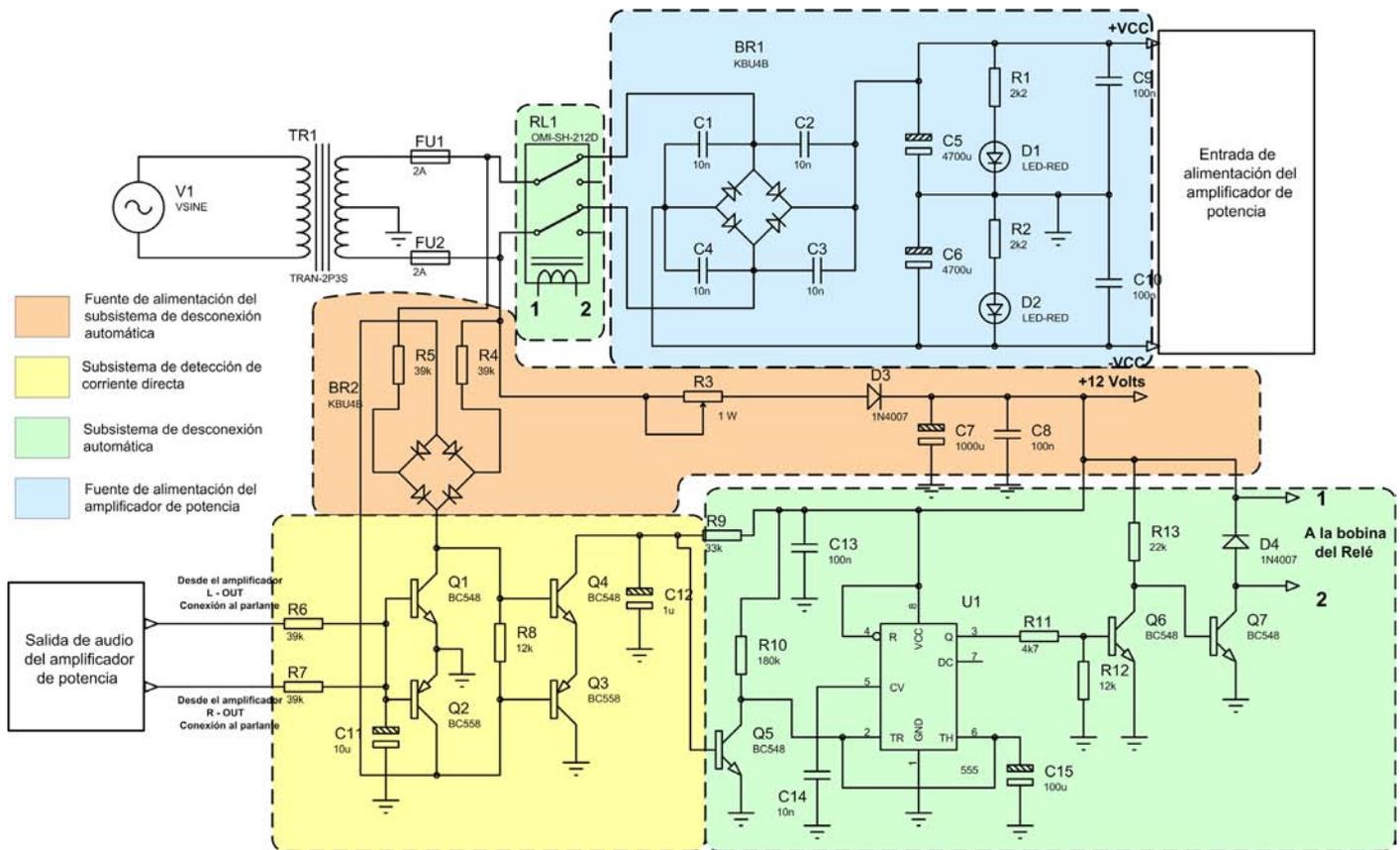



FIGURA 2: Esquema empleado durante la explicación del diseño. (Descargalo en www.ucontrol.com.ar)

.La alimentación del circuito

A través de los fusibles FU1 y FU2, la tensión de corriente alterna (CA) obtenida del transformador, toma dos caminos muy diferentes. Por un lado, atraviesa los contactos Normalmente Cerrados (NC) de RL1 hacia la fuente de alimentación del amplificador de potencia; y por otro, se conecta a la fuente de alimentación del sistema de protección.

Una vez atravesados los contactos NC de RL1, nos encontramos con el puente rectificador BR1, los capacitores electrolíticos C5 y C6, y los indicadores LED, que son opcionales. Completan este subsistema, los capacitores de desacoplo de 100nF C9 y C10, que se utilizan para suprimir el ruido de alta frecuencia. Hasta aquí tenemos una fuente de alimentación simétrica, que se utilizará para alimentar el amplificador de potencia.

A través de R3, D3, C7 y C8, obtendremos una tensión cuyo valor debe ser 12 V, la cual podremos fijar mediante el potenciómetro R3. Esta tensión alimenta al subsistema de desconexión automática.

Como habíamos visto antes, la tensión de CA obtenida de TR1 atraviesa también R4 y R5, hacia BR2 quién se encargará de rectificarla y entregarnos una tensión de aproximadamente 2V. Esta tensión alimenta el subsistema de detección de corriente directa (CD).

La tensión de salida debe mantenerse dentro de los parametros normales para que trabajen todos los eslabones de la cadena de audio.

.Tras el camino de la protección

Cuando un amplificador de audio funciona normalmente, existe en su salida, una tensión variable en el tiempo y en función de los sonidos que se estén escuchando. Por lo general la misma debe mantenerse dentro de los parámetros normales de funcionamiento para que trabajen todos los eslabones de la cadena de audio, de forma armoniosa, estable y en los márgenes de seguridad deseados.

Si la tensión de salida del amplificador, se hace cada vez más alta producto del aumento de volumen de audio, nos estaremos acercando peligrosamente a que la misma esté muy próxima o iguale a Vcc, tanto en su polaridad positiva como negativa. En estos casos, el sistema de protección debe entrar en acción y desconectar la alimentación del amplificador de potencia.

Para saber si a la salida del amplificador existe una tensión de CD peligrosa, R6 y R7 toman una muestra de ambos canales de salida de audio. Q1 y Q2 trabajan normalmente en sus regiones activa y de corte. Si la tensión en R6 o R7 crece demasiado, pueden llevar a Q1 o Q2 al estado de saturación, desencadenando el proceso de desconexión del amplificador.

El capacitor C11 se encarga de retardar el momento de conducción de Q1 o Q2, ya que pueden pre-

sentarse picos de tensión provocados por alto volumen y no por problemas o fallas en la etapa de salida. Pero una vez que actúa cualquiera de los dos transistores, según la polaridad de la tensión incidente sobre ellos, el potencial presente en su colector, se drenará a tierra, y el transistor entrará en su región de saturación. Este hecho será detectado por el circuito formado por R8, Q4 y Q3 quienes entrarán en saturación obteniéndose un estado lógico bajo en el colector de Q4 que está alimentado por R9. Hasta aquí tenemos todo el subsistema de detección de tensión de CD a la salida del amplificador.

Al saturarse Q3 y Q4, provocan que Q5 entre en corte y se eleve el estado lógico de los pines 2 y 6 del CI 555, según la constante de tiempo del conjunto R10-C15. Una vez que en los extremos de C15 se hayan superado los 2/3 de la tensión de alimentación del CI 555, este cambiará de estado en su pin 3 de salida, provocando el corte de Q6 y como consecuencia la saturación de Q7. De esta forma se activará RL1, quien interrumpirá el suministro de energía a la fuente de alimentación de nuestro amplificador. Este conjunto de componentes conforman el subsistema de desconexión automática.

Una vez que el sistema de protección se ha activado, permanecerá en este estado, interrumpiendo la reconexión del amplificador. Esto se logra gracias a que la alimentación del subsistema de desconexión automática, se toma a través de R3, antes de los contactos del relé.

Básicamente este es el funcionamiento del sistema de protección, el que podremos acondicionar fácilmente a nuestros sistemas de sonido, con un poco de imaginación e ingenio, dotándolo de las prestaciones de los equipos de primera línea.

//Diego Márquez García-Cuervo

fundamentos de la transmisión sincrónica

Una visión general de qué es, para qué sirve y cómo se utilizan las Transmisiones Síncronas. Para acercarnos de forma clara a qué es una transmisión Síncrona vamos a utilizar una forma indirecta de atacar las cosas, comenzando por su antagonista por naturaleza: la transmisión Asíncrona

Primero démosle un vistazo a su propio nombre y veamos qué significa esa palabreja de **Asíncrona**. Etimológicamente significa exactamente “sin reloj” o sea que no hay ninguna señal que marque los tiempos en que los datos deben leerse o están disponibles.

Esto significa que en una transmisión asíncrona tanto la información transmitida como los tiempos en que ésta debe leerse son solo uno y todo va junto. El mejor ejemplo de este tipo de transmisión es la transmisión serie RS-232. En esta forma asíncrona de transmitir información binaria cada bit es representado por un estado Alto o Bajo de la línea de transmisión durante un tiempo predeterminado. Este tiempo debe ser siempre el mismo, dentro de los márgenes de tolerancia normales y que son de aproximadamente de un 2% del valor nominal.

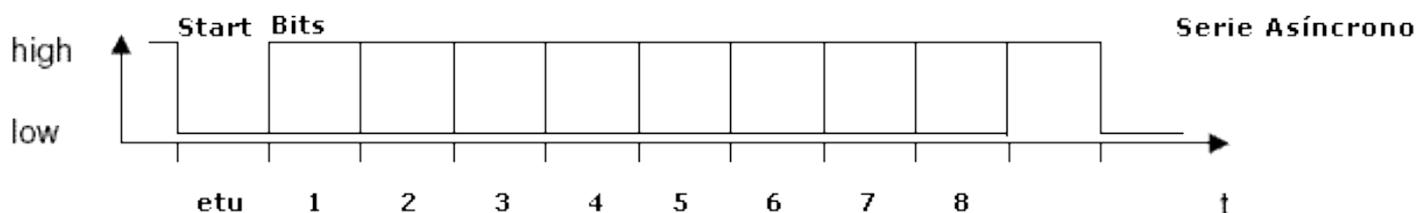
Fijaos por tanto que esto de **Asíncrono** no significa sin tiempo sino bien al contrario: significa con tiempos perfectamente definidos y acordados de antemano ya que de otra forma no habría manera de poner de acuerdo al emisor y al receptor en cuanto a cuando está disponible cada bit para su lectura.

> **Asíncrono no significa sin tiempo sino bien al contrario: significa con tiempos perfectamente definidos**

El sistema asíncrono funcionaría entonces así: En cuanto el receptor detecta el primer cambio de estado, una línea que pasa de alto a bajo por ejemplo en el RS232, sabe con total seguridad que tras cierto número de microsegundos transcurridos tendrá disponible el primer bit transmitido por el emisor, y tras otro igual número de microsegundos tendrá el segundo bit y ... así hasta el último bit que debe recibir.

Se detecta el primer flanco de bajada y a partir de ahí solo debe mirar, cada plazo de tiempo acordado, en qué estado está la línea de transmisión, si alto o bajo, para asignar ese valor a cada uno de los bits a recibir.

De esta forma cuando decimos que una comunicación RS232 es a 8 bits y a 9600 baudios lo que estamos diciendo es que vamos a recibir 8 estados consecutivos de la línea de transmisión, separados cada uno de ellos 1/9600 segundos, o sea un estado cada 104 microsegundos, siendo el primero el estado que tenga tras los primeros 104 microsegundos transcurridos desde el primer flanco de bajada.

A 19.200 baudios el “tiempo” de cada bit será la mitad, 52 microsegundos, y a 4.800 baudios será el doble o sea 208 microsegundos. A esta unidad de tiempo la conocemos como el **ETU** de una transmisión, iniciales de **Elementary Time Unit** (Unidad de Tiempo Elemental). Abajo podemos ver una representación gráfica de esto que estamos tratando, la transmisión Asíncrona de un byte compuesto por 8 bits (un típico **8N1** a 9.600 baudios).

Un byte a 9600 baudios.

Una conclusión a la que podemos llegar después de expuesto todo esto sobre la transmisión Asíncrona es que es imprescindible saber a priori a qué velocidad vamos a recibir los distintos bits para ajustar nuestra rutina de recepción a dicha velocidad y mirar así la línea de transmisión en su momento justo, ni antes ni después, para recibir cada uno de los bits en el momento en que realmente les corresponde. Cualquier error en el cálculo dichos tiempos puede hacernos leer "bits fantasmas", debido a que leemos dos veces un mismo bit o porque nos salteamos alguno de ellos.

Y por fin llegamos a nuestra Transmisión Síncrona de datos.

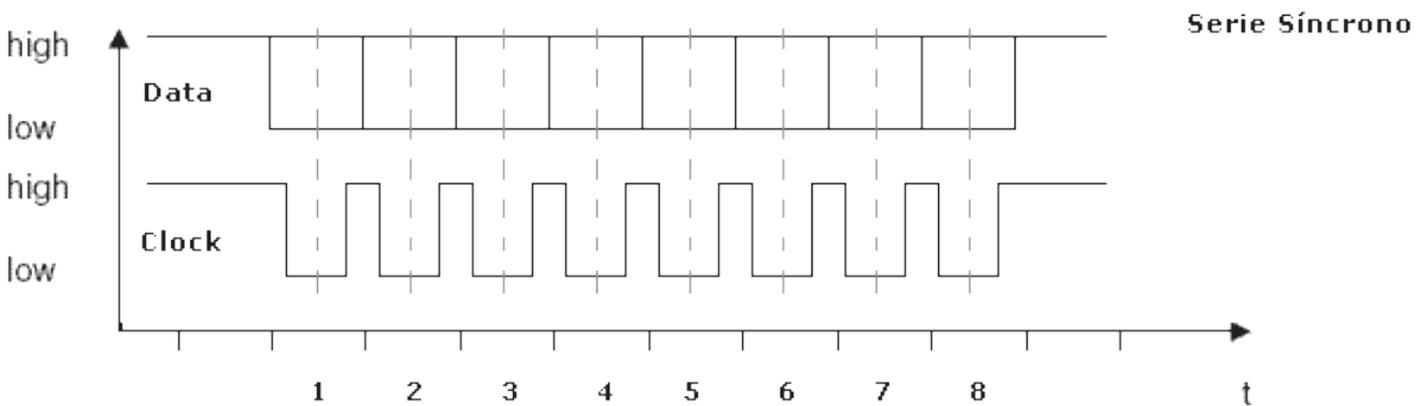
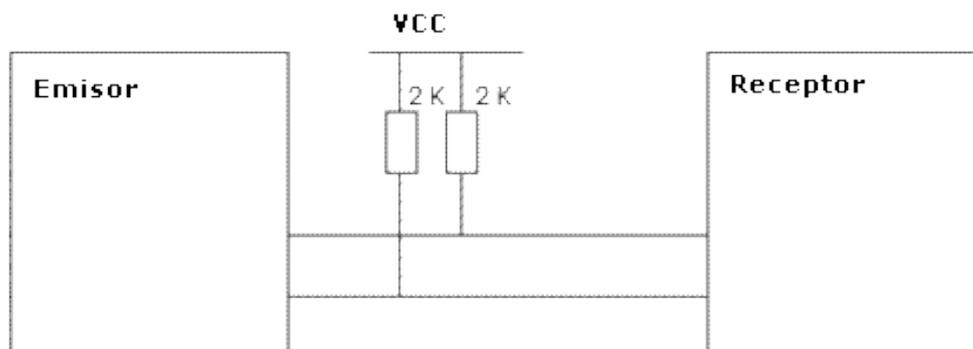
Síncrono significa **"con reloj"** y exactamente eso es lo que necesitamos, un reloj (o dicho en inglés un **Clock**). La transmisión síncrona necesita de dos líneas, una de datos sobre la que se van a representar los distintos estados de los bits a transmitir y una de reloj donde vamos indicando cuando está disponible cada bit en la línea de datos. Esta línea de reloj es la de "sincronización" entre ambos dispositivos, el emisor y el receptor de la transmisión.

De esta forma, una transmisión síncrona consiste exactamente en poner el estado de un bit en la línea de datos, generar un pulso de subida y uno de bajada en la línea del reloj, poner otro estado de bit en los datos, volvemos a dar un pulso de subida y bajada en la del reloj... y así hasta completar el número de bits que deseemos transmitir.

Esta forma de transmisión tiene una clara ventaja, y es que no es necesario poner de acuerdo en velocidad alguna a emisor y receptor de la transmisión. El emisor coloca su bit y genera el pulso en el reloj, el receptor detecta el reloj y mira el estado del bit, y así uno tras otro, a cualquier velocidad, a distinta velocidad cada bit, a toda la velocidad posible. Hay pulso significa hay dato, leo y a esperar otro pulso, más lento o más rápido es irrelevante solo es importante aquello de pulso-dato y a empezar de nuevo.

La única limitación es que al receptor le debe dar tiempo a leer el estado de cada bit tras detectar el pulso de reloj antes de que aparezca un nuevo pulso.

Notad que en estos ejemplos estamos utilizando la "lógica negativa" es decir que detectamos los pulsos estando la línea en alto cuando cae a bajo, o sea recibiendo primero un flanco de bajada y después uno de subida para conformar un pulso.

Todo lo que estamos tratando sería exactamente igual con los pulsos al revés, en "lógica positiva" con el flanco de subida primero y el de bajada después. Esta configuración con las líneas en alto y dando pulsos negativos es la más utilizada debido a la estabilidad y resistencia al "ruido" que tienen. Se consigue conectando una resistencia a **VCC** para que mantenga la línea a estado alto y nuestro emisor genera los pulsos poniendo la línea a **GND**. El receptor está constantemente recibiendo el estado alto y detecta cada pulso cuando pasa a bajo. Este es el concepto de **Pull-Up.**

Pulsos de reloj y datos en una transmisión síncrona.

Resistores "Pull-Up" en las líneas de comunicación.

Las funciones que vamos a implementar son: Transmite_Bit_Clock_and_Data y Transmite_Byte_Clock_and_Data

Ahora vamos a ver cómo podemos implementar una simple comunicación síncrona en C. Utilizaremos el compilador **CCS**, sobre el que incluiremos un tutorial completo en próximos números de la revista.

Las funciones para transmitir de forma síncrona que vamos a implementar son dos: **Transmite_Bit_Clock_and_Data** y **Transmite_Byte_Clock_and_Data**. La primera de ellas coloca el estado de un bit en la línea **Data** y genera un pulso en la de **Clock**. La segunda se encarga de extraer, bit a bit, el contenido de un byte (8 bits) y llamar a la función anterior.

Para las funciones de recepción síncrona vamos a usar el recurso de la Interrupción Externa de los PIC's, eligiendo estratégicamente el PIN del reloj (CLOCK) de forma que tengamos disponible una de estas interrupciones.

La interrupción externa la configuramos para detectar los flancos de bajada (ver recuadro sobre la "lógica negativa"). De esta forma cada vez que se dispara la interrupción sabemos que tenemos disponible un bit en la línea de los datos (DATA). Lo recogemos sobre nuestro recByte y contamos uno más. Cuando lleguemos a 8 bits recogidos tenemos nuestro Byte completo y podemos indicarlo convenientemente poniendo a uno el flag reccomplete.

Cuando en main detectamos este reccomplete lo monitorizamos por el puerto serie y reiniciamos todo para recoger el siguiente byte.

Código:

```
 #define OUT_CLOCK PIN_B0
 #define OUT_DATA  PIN_B1

 void Transmite_Bit_Clock_and_Data(int1 bit){
// Coloca Data
 if( bit==0){ output_high(OUT_DATA); }
              else{output_low(OUT_DATA); }

// Genero pulso en Clock (500 microsegundos ó
2 Khz)
delay_us(250);
output_low(OUT_CLOCK);
delay_us(250);
output_high(OUT_CLOCK);
}

void Transmite_Byte_Clock_and_Data(char c){
int8 i;
int1 b;
for(i=0;i<8;i++){
b = bit_test(c,i);
Transmite_Bit_Clock_and_Data(b); }
}
```

Código:

```
#define IN_CLOCK PIN_B0
#define IN_DATA  PIN_B1

char recByte=0;
int8 nextBit=0;
int1 reccomplete=0;

// INTERRUPCIÓN por EXTO Clock CK (Data - Clock)
--------
#int_ext
ext_isr() {
int1 bit;

bit=!input(IN_DATA);
bit_clear(recByte,nextBit);
if(bit==1) bit_set(recByte,nextBit);
if(++nextBit=8){
nextBit=0;
reccomplete=1; }
}

 // MAIN -------------------------------------
void main(void){
ext_int_edge(0,H_TO_L);
enable_interrupts(int_ext);
enable_interrupts(global);

do {
// Lectura Completa
if(reccomplete==1){
readcomplete=0;
putc(recByte); }
         } while (TRUE);
   }
```

Lógica Negativa *por Reinier Torres Labrada*

Una gran ventaja de la lógica negativa es que facilita el arbitraje de buses compartidos, como I2C, ya que cualquier dispositivo que ponga en estado bajo la línea de Reloj se está apoderando del bus. No ocurre así con la lógica positiva, puesto que en ese caso cualquier dispositivo que pusiese la línea de reloj en estado bajo inhabilitaría a todo dispositivo que tiene esta línea en estado alto.

Es decir que no importa cuántos elementos en el bus tengan su línea de reloj en estado alto, basta que una ponga un estado bajo para que toda la línea quede en este estado. Es fácil ver que en el caso contrario esto no ocurre así.

De hecho la lógica negativa no es desde el punto de vista del consumo la mejor opción: si se observa cualquier circuito integrado se verá que los estados lógicos bajos tienen corrientes de circulación mayor, pero para cosas como esta es lo que mejores resultados da. Por ejemplo un bus arbitrado o con alguna lógica de contención de buses, lo que hace es precisamente poner una entrada a estado lógico bajo, hasta que se haga con el control del BUS. Es simple pero efectivo.

I2C, SPI, Ethernet y otros por el estilo trabajan de esa forma.

construye tus propios PCB

Todo aficionado a la electrónica tarde o temprano se topa con la necesidad de fabricar sus propios circuitos impresos o PCB (por “Printed Circuit Board”).
En general, esta tarea es vista como algo muy complicado, cuando en realidad se trata de una labor que fácilmente podemos llevar a buen término.

A lo largo de este artículo aprenderemos a desarrollar placas de circuito impreso para nuestros proyectos, utilizando para ello elementos de fácil adquisición, y que tendrán un acabado muy profesional. Sobre ellas montaremos los componentes de nuestros proyectos de electrónica, y si somos prolijos, resultaran casi indistinguibles de un PCB fabricado por métodos industriales.

Emplearemos el método conocido como “Método de la plancha”, llamado así por que se utiliza una plancha convencional (de las empleadas para planchar la ropa) para transferir el tóner de una impresión LASER o fotocopia a una placa de circuito impreso virgen.

.Elementos Necesarios

Para encarar la realizaron de un PCB, deberemos tener a mano una serie de elementos que resultan indispensables para dicha tarea.

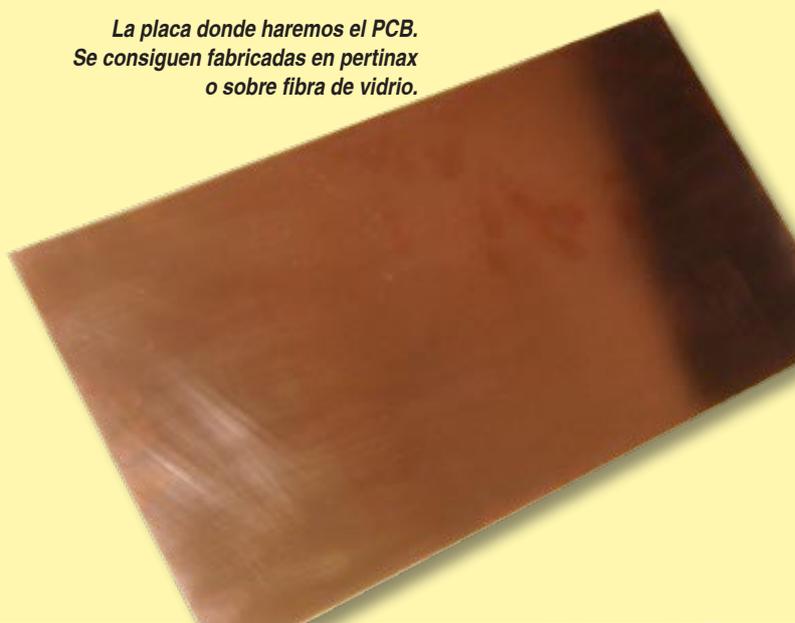
•El diseño o dibujo de nuestro circuito impreso. Que podremos realizar utilizando algún programa especializado, como Eagle, Orcad, o hasta con el mismísimo Microsoft Paint incluido en todas las versiones de Windows.

•El papel. Los mejores para el procedimiento descrito son aquellos utilizados para imprimir revistas o catálogos, consiga alguno que sea delgado, pero no demasiado, porque el calor de la impresora puede deformarlo y con ello atentar contra el resultado final. No tiene por que ser papel virgen, puede estar impreso. La práctica y el uso de distintos papeles le llevará a encontrar el mejor para usted.

•Impresora LASER. Luego necesitamos una impresora LASER o una fotocopiadora. Como veremos más adelante, el tóner de la impresión es el que formará las pistas de nuestro PCB. Si no disponemos de una, podemos usar cualquier impresora, y llevar el impreso a una fotocopiadora y hacer una copia. Las fotocopias también son hechas mediante tóner, por lo que gracias a este procedimiento nos haremos de un original para nuestro PCB.

•La placa virgen o PCB. También debemos comprar en una tienda especializada en componentes electrónicos (donde compramos los demás componentes para nuestros circuitos) una placa de PCB virgen, del tamaño adecuado para nuestro proyecto. Estas placas generalmente se consiguen fabricadas en pertinax o sobre fibra de vidrio. Cualquiera de las dos sirve.

La impresión debe ser láser. El tóner de la impresión es el que formará las pistas de nuestro PCB.

La placa donde haremos el PCB. Se consiguen fabricadas en pertinax o sobre fibra de vidrio.

•Limpiadores. Algún limpiador de uso doméstico y un poco de lana de acero, de los mismos que se utilizan para lavar los cacharros de cocina, serán los elementos que nos permitan dejar bien limpia y desengrasada la superficie de la placa.

•La plancha. Es la herramienta fundamental en el procedimiento, ya que nos permitirá transferir el circuito impreso en papel al cobre de la placa. Dominar la técnica requiere tiempo y práctica, pero una vez que lo consiga, podrá hacer casi cualquier tipo de PCB de poca complejidad.

•El decapante. Lo utilizaremos para retirar el cobre de la placa que no forma parte de las conexiones del circuito. Un cuarto litro de percloruro férrico (o cloruro férrico, el nombre puede variar de un lugar a otro, pero se trata de la misma sustancia), será suficiente para comenzar.

•Herramientas y accesorios. Una agujereadora, con una broca de 1.00 mm y otra de 0.75 mm, un recipiente plástico en el que entre la placa, uno metálico en el que entre el recipiente plástico, y una sierra de cortar metales complementan el conjunto de elementos casi indispensables para llevar a buen término la confección de nuestras PCB.

Necesitarás una plancha como ésta.

El percloruro férrico será el encargado de eliminar el cobre sobrante.

Un "Dremel", ideal para agujerear los PCB.

.Impresión del circuito

Una vez que tenemos listo el dibujo de nuestro circuito impreso, ya sea porque lo hayamos realizado nosotros en el ordenador o que lo hayamos bajado de Internet, debemos transferirlo al papel.

En este punto, debemos tener en cuenta un par de consejos para que el resultado final sea óptimo. En primer lugar, la escala del dibujo debe ser la adecuada para que cuando vayamos a montar los componentes en nuestro PCB, las medidas coincidan. Por ejemplo, la separación estándar entre dos pines consecutivos de un circuito integrado es de 0.1 pulgada (2,54 mm). Si nos atenemos a esto, no tendremos problemas. Todos los archivos PDF correspondientes a los PCB de los proyectos que puedes descargar desde www.ucontrol.com.ar están en la escala correcta, listos para imprimir.

En segundo término, como veremos más adelante, al transferir el dibujo del papel al cobre la imagen quedará invertida, como si la viéramos en un espejo, así que debemos tener esto en cuenta al dibujarlo en el ordenador para no terminar con una imagen invertida en el PCB. No es conveniente imprimir nuestro circuito con la opción de economía de tinta activada, ya que necesitamos una buena cantidad de tóner en la copia, dado que es el que se va a transferir al cobre.

Tener en cuenta: la escala del dibujo debe ser la adecuada para que cuando vayamos a montar los componentes las medidas coincidan. Al transferir el dibujo al cobre, éste quedará espejado.

Si nuestra impresora no es láser, como dijimos antes, llevaremos nuestra impresión a una fotocopiadora y haremos una copia de ella, cuidando que la escala sea exactamente 1:1 (no todas las fotocopias son idénticas al original) y que la copia no presente rayas o cortes, ya que de ser así, estas imperfecciones se transferirán al PCB. Si no estamos conformes con la calidad de la fotocopia, hagamos sacar otra hasta que veamos que no tiene defectos.

Respecto del papel a utilizar, los mejores resultados los he obtenido utilizando papel ilustración, que es un papel de una calidad mayor al de resma común, con un grano más fino y ligeramente satinado. Incluso hay aficionados que emplean el papel "satinado" que obtienen de páginas de las revistas viejas, con muy buenos resultados.

Como puede verse, lo mejor es hacer varias pruebas hasta encontrar el tipo de papel adecuado antes de comprar grandes cantidades de un tipo determinado.

.Transferencia al cobre

En esta etapa del proyecto, debemos transferir el tóner del papel al cobre, para lo cual utilizaremos el calor que nos brindará la plancha.

Una vez cortada la placa virgen a las medidas de nuestro PCB con la sierra para metales, comenzaremos la limpieza concienzuda de la placa de circuito impreso virgen, para que quede libre de suciedad, y grasa. Utiliza-

remos para ello el polvo limpiador y la lana de acero, que debe ser lo más fina posible para que no queden rayas profundas. Algún agente químico puede resultar útil, como por ejemplo un limpia metales y un trapo en lugar de la lana de acero. Podemos probar distintos métodos, de acuerdo a los elementos que tengamos a mano. Mientras llevamos a cabo esta tarea, podemos aprovechar y enchufar la plancha para que vaya tomando temperatura.

Es importante aclarar que algunas marcas de limpiametales depositan sobre el cobre una película protectora, para evitar que el oxigeno presente en el aire oxide el metal, manteniéndolo brillante durante mucho tiempo. Pero esa misma capa protectora puede hacer que el percloruro sea incapaz de atacar el cobre, así que también aquí debemos probar entre distintas marcas. De todos modos, el uso de la lana de acero da excelentes resultados con poco trabajo.

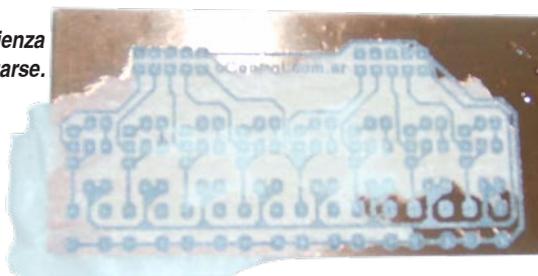
Lana de acero fina, o "Virulana".

Una vez que el cobre está limpio, alinearemos sobre el PCB el papel con la impresión que hemos hecho, con el dibujo de las pistas hacia el cobre (debemos ver la parte sin imprimir), de manera que cuando apliquemos calor, el toner se funda y se transfiera al cobre.

Una vez colocado el papel sobre la placa, lo podemos fijar con cinta adhesiva por el otro lado de la placa, evitando con ello que el papel se corra durante el proceso de planchado y se estropee el circuito.

Con la plancha bien caliente "planchamos" la hoja durante uno o dos minutos, aunque este tiempo puede variar de acuerdo al tipo de tóner y la temperatura exacta de la plancha, con lo que casi todo el toner se habrá pegado a la cara de cobre del PCB.

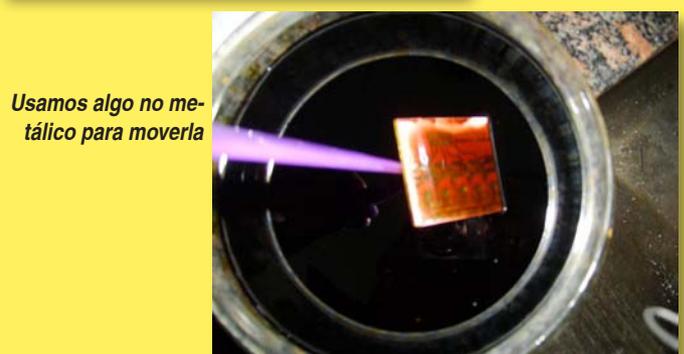
Para remover el papel del PCB sumergimos la placa en agua del grifo durante unos 5 o 10 minutos (depende del tipo de papel), y luego con los dedos o un trapo mojado iremos desmenuzando el papel con cuidado hasta eliminarlo por completo del PCB. En este momento deberíamos tener la placa con el dibujo listo. Solo resta asegurarnos que todas las pistas y nodos se hayan calcado correctamente, y que no hayan quedado pedacitos de papel que puedan evitar la acción del percloruro, dando lugar a cortocircuitos en nuestro PCB terminado.

El papel comienza a desmenuzarse.

.Eliminado el cobre no deseado

El proceso que llevaremos a cabo a continuación tiene como fin eliminar todas las zonas de cobre que sobran de nuestra placa virgen, es decir, las que no están cubiertas por el tóner.

Para ello, pondremos algo de agua en el recipiente metálico que mencionamos al principio de la nota, y dentro de él ponemos el recipiente plástico con el percloruro. Ponemos todo el conjunto en una hornilla, a fuego mínimo, como para que el agua caliente a unos 40 ó 50 grados el percloruro que se encuentra en el recipiente plástico. El calor hará las veces de un catalizador positivo, provocando que el percloruro ataque con mayor velocidad las zonas de cobre desnudo.

Ponemos la placa dentro del percloruro, y esperamos unos 5 ó 10 minutos hasta que el cobre que esta sin cubrir desaparezca. A veces, da buen resultado mover suavemente la placa durante este tiempo, para evitar que el percloruro que ya se combinó con el cobre se deposite sobre la placa y actúe como un "aislante" que evita el contacto del PCB con el percloruro sin combinar.

Ponemos la placa dentro del percloruro.

Usamos algo no metálico para moverla

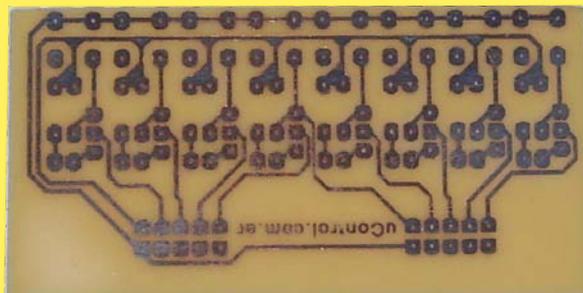
¡Listo! Ya no queda cobre por eliminar.

Este proceso debe ser realizado en un lugar bien ventilado y lejos de los alimentos o zonas donde se preparen o almacenen, debido a los vapores ligeramente tóxicos que se desprenden del proceso y a que el percloruro férrico es una sustancia tóxica, lo recomendable es hacer el proceso en el patio o balcón, al aire libre. De igual modo debe tener cuidado sobre el lugar donde almacena esta sustancia y al finalizar todo lo recomendable es lavar bien todo con agua. No tire el percloruro deteriorado por el uso por el caño del desagüe porque puede estropear las tuberías.

El percloruro que utilizamos podemos guardarlo para otra placa, ya que en general, y dependiendo de la superficie del PCB realizado, se puede emplear varias veces. Cuando notemos que el tiempo necesario para acabar el trabajo es demasiado largo (25 o 30 minutos) será el momento de comprar otro bidón de percloruro.

Con la placa ya libre de percloruro, utilizamos nuevamente la lana de acero con el polvo limpiador para remover todos los restos de tóner que hay sobre el PCB, y ya tendremos nuestro PCB casi listo, restando solamente efectuar los agujeros para los componentes.

Para agujerear la placa, usaremos el taladro y las brocas, cuidando que los agujeros queden centrados sobre los pads del PCB, y que el diámetro de los mismos sea el adecuado para los terminales de los componentes que usaremos. Agujeros demasiado grandes o pequeños impedirán que el resultado final tenga excelente calidad.

Lavamos el PCB con abundante agua.

El PCB limpio, listo para agujerear.

PCB agujereado. Solo resta montar los componentes.

.Consejos Finales

Para terminar, vamos a mencionar algunos consejos para que nuestro trabajo resulte más prolijo y satisfactorio.

- Para dibujar el PCB en el ordenador, se puede utilizar programas comerciales de uso general, como Corel Draw, Autocad, o incluso el sencillo Paint, siempre cuidando de que las dimensiones de los objetos que dibujemos sean las correctas.
- No es mala idea imprimir en un papel común una copia y sobre ella “medir” los componentes para ver si las distancias entre pines son las adecuadas. Existen programas específicos para la realización de PCB, como PCBWizard, Pad2Pad, FreePCB o Spicycle, algunos de ellos disponibles en forma gratuita. De todos modos, si nuestro proyecto no es muy complejo, se puede realizar el diseño del PCB tranquilamente sin necesidad de software especializado.
- Cuando hagamos nuestros dibujos, podemos “pintar” los espacios que quedan entre componentes o entre pistas, para que sea menor la superficie que debe atacar el percloruro. Esto hará que el tiempo necesario para llevar a cabo la tarea sea menor, y que el percloruro nos sirva para un mayor número de placas.
- El liquido sobrante debemos guardarlo bien tapado, en un lugar fresco y si es posible que no esté expuesto a la luz del sol directa, para evitar que se degrade prematuramente.
- Por favor, sea responsable al eliminar los restos de los productos químicos empleados.

uso práctico del PIC12F675

primera parte: temporizador para WC

Esta es una sección dedicada especialmente al microcontrolador de Microchip PIC12F675 que se inaugura con un automatismo práctico. Usaremos este componente como temporizador, a la vez que veremos cómo controlar la línea de 220AC.

El PIC12F675 es un microcontrolador, que como todos sus hermanos incorpora una serie de periféricos en su interior. Estos módulos son circuitos especializados. En sucesivos artículos iremos viendo con que módulos cuenta y como hacer uso de ellos, lo que nos dará una idea de sus posibles aplicaciones.

Al igual que el resto de los PIC´s, están diseñados siguiendo la arquitectura Harvard, donde la memoria de datos está separada de la de programa. Y también son RISC (Reduced Instruction Set Computer), por lo que tienen un juego reducido de instrucciones, compuesto por solo 35 de ellas. Este valor puede variar, dependiendo de la familia a la que pertenezca un PIC en particular

El 12F675 pertenece a la familia de micros “enanos” de Microchip. Tiene solo 8 pines, 1024 Word de memoria Flash (también llamada memoria de programa), 64 Bytes de memoria RAM, 128 Bytes de memoria EEPROM y se puede conseguir en distintos encapsulados. Es económico, y aunque solo tiene 6 pines aprovechables, se le puede sacar buen partido para usarlo como descarga de procesos de un microcontrolador más grande o como “cerebro” de distintos automatismos simples.

El 12F675 pertenece a la familia de micros “enanos” de Microchip. Tiene solo 8 pines, 1024 Word de memoria Flash (también llamada memoria de programa), 64 Bytes de memoria RAM, 128 Bytes de memoria EEPROM y se puede conseguir en distintos encapsulados.

En este proyecto utilizaremos los siguientes módulos internos del PIC:

-El reloj interno trabajando a 4Mhz.

-El modulo de entradas y salidas (I/O).

-El modulo Watchdog (Perro Guardián), este modulo lo nombraremos como WDT.

-El Timer1.

Al utilizar su generador de reloj interno nos ahorramos el cristal de cuarzo y sus dos condensadores asociados.

Con su modulo I/O programado especialmente podremos setear la entrada de 220AC, controlar el encendido de dos testigos indicadores (diodos LED) y además activar un rele.

Los pines del modulo I/O se pueden configurar como entrada o salida digital, exceptuando el pin 4 (llamado GP3), que solo funciona como entrada. Como no lo usamos, lo conectaremos al plano de masa (GND). El resto de los pines que queden sin usar serán configurados como salidas y los dejaremos sin conexión o “al aire”. Y como seguramente habrán visto en la hoja de datos, las patitas del micro además del numero de orden del pinout también tiene nombre especifico (“GP” seguido de un número). En este caso hay que tener especial cuidado ya que en esta familia se les da un nombre distinto al resto de familias de microprocesadores PIC.

Utilizando el modulo WDT nos aseguramos que se generará un RESET interno del PIC en caso de que se produzca algún bloqueo del programa causado por efectos desconocidos (normalmente ruido eléctrico). El modulo WDT necesita contar tiempo y lo hace reservándose para él el Timer0. Cuando el Timer0 se desborda, el WDT nos genera un RESET, así que para que esto no ocurra durante el funcionamiento normal del programa tenemos que borrar el WDT. Esto a su vez reiniciará el Timer0. Ésta operación tenemos que procurar hacerla antes que termine de contar el tiempo que se le programó.

Para refrescar el WDT se siguen tres sencillas reglas, y son las siguientes:

1) Se comenzará a refrescarlo en la rutina principal del programa.

2) A lo largo del programa se refrescará el menor número de veces posible.

3) Siempre se evitará refrescarlo en la rutina de interrupciones (si las hay).

El Timer1 lo utilizaremos para generar unas bases de tiempos que nos servirán para controlar el parpadeo de los led, refrescar las salidas y controlar cada cuanto tiempo se ejecutan las rutinas del programa en general.

.Objetivos:

El propósito general de este automatismo es el de controlar un ventilador, usado como extractor para el cuarto de baño. El circuito tiene que cumplir tres requisitos básicos:

1º Cuando está en reposo, el automatismo no puede tener consumo alguno.

2º El circuito será controlado por un PIC "enano", el PIC12F675

3º Deberá ser capaz de controlar un ventilador de 220AC o 12V/<180mA.

La función que realizará el circuito se llevara a cabo de la siguiente forma:

En el estado inicial, con el interruptor que controla la lámpara principal del baño en la posición de "abierto", el rele se encuentra en estado de reposo, tal como se ve en el esquema.

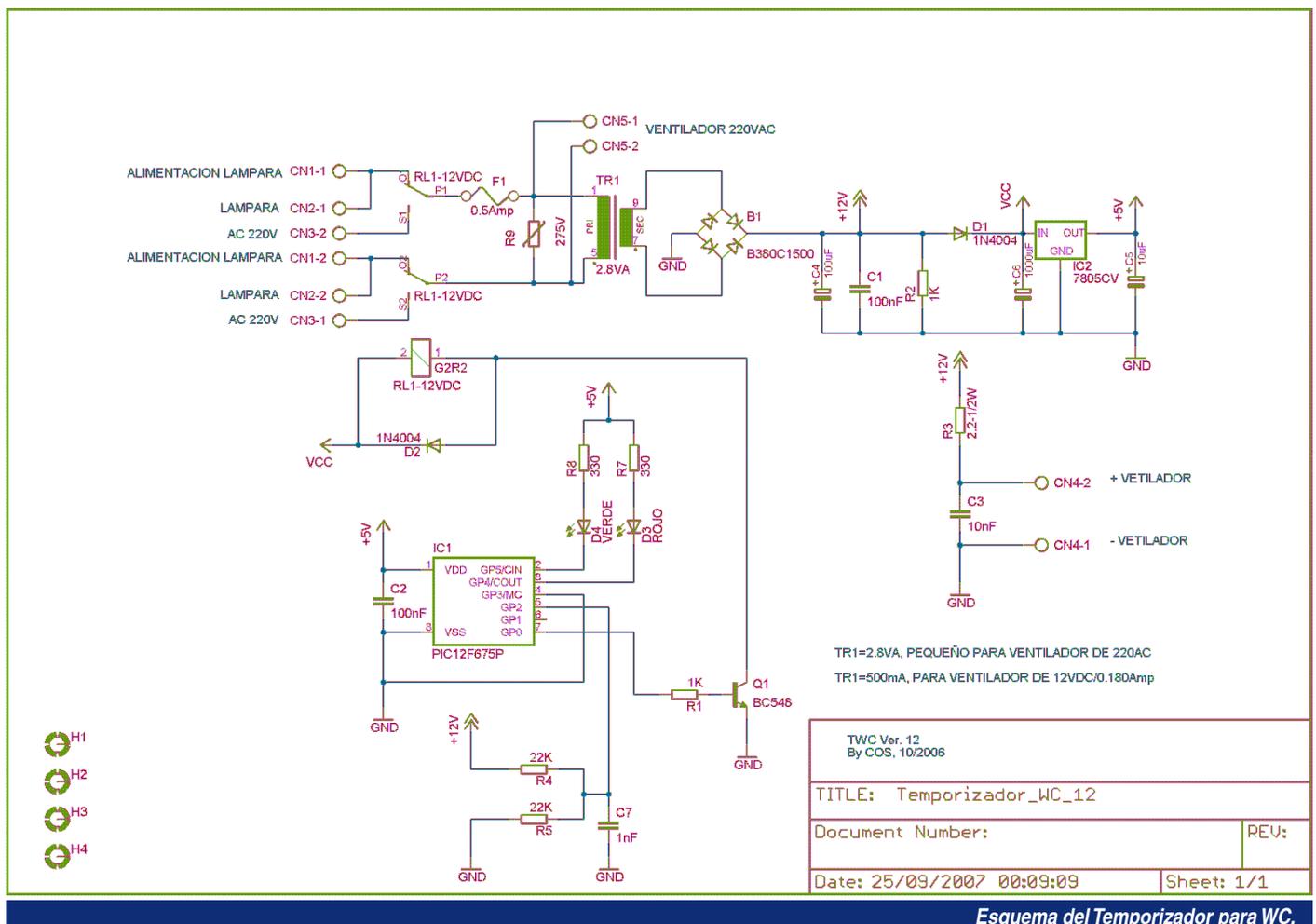
Al energizar la lámpara mediante el interruptor de la pared, queda también alimentado el circuito. El rele no se activa hasta pasados 30 segundos, por lo que durante ese tiempo el circuito dependerá del interruptor de encendido de la lámpara del WC.

Pasados estos primeros 30 segundos queda en modo de lectura de la alimentación por la entrada GP2. Cuando el programa detecta que la alimentación cae, activa el rele aprovechando la carga de C6. De este modo todo el circuito pasa a estar alimentado directamente de la línea de 220VAC a través de los contactos del rele, comenzando un periodo de temporización de 2 minutos de duración.

Cuando este tiempo a transcurrido, el PIC corta la alimentación del rele, y este pasa al estado de reposo. Si el interruptor de la pared que controla la lámpara está abierto, el circuito pierde la alimentación y todo queda sin energía. En caso de que el interruptor siga cerrado, el circuito espera 1 segundo y luego pasa de nuevo al modo de seteado de la alimentación, quedando en este estado de monitorización de la red eléctrica esperando a que esta desaparezca para comenzar una nueva temporizacion.

.El funcionamiento del circuito:

Los conectores CN1 y CN2 se utilizan para alimentar el circuito desde la toma de la lámpara y para alimentar la lámpara respectivamente, para no tener que hacer modificaciones en la instalación eléctrica de la casa.

Esquema del Temporizador para WC.

El conector CN3 es una toma de corriente de 220 AC directa de red.

Cuando se activa el rele, sus contactos suministran la energía tanto al circuito como al ventilador 220VAC conectado al CN5, quedando todo alimentado desde la red de forma directa. R9 es un varistor de 275V, y es un estupendo filtro de red.

Después del puente rectificador pueden verse los condensadores C4 y C1, que son el filtro de la fuente. Siguiendo con el circuito del ventilador de CC, es muy importante colocar el condensador C3/10nF para disminuir el ruido que genera el motor eléctrico y que es inyectado en la fuente. Este condensador tiene que estar lo mas cerca posible de los contactos del CN4.

Para que el PIC pueda comprobar la perdida de alimentación se usa un divisor de tensión compuesto por R4 y R5, mas la capacidad C7 para dar algo de estabilidad a la señal. También nos aprovechamos del diodo interno limitador que tiene este pin y que nos limita la tensión (5.5v) que le llega desde el exterior. D1 y C6 nos permiten prolongar la alimentación del PIC frente a caídas de la alimentación. C4 tiene que ser de menor capacidad que C6, y se añade R2 que ayuda a la descarga de C4, para hacer que en el caso de ausencia de tensión el PIC pueda detectar esta caída antes de quedarse sin alimentación, sobre todo en el caso de no utilizar un ventilador de 12VDC.

Para que el PIC pueda comprobar la perdida de alimentación se usa un divisor de tensión compuesto por R4 y R5, mas la capacidad C7 para dar algo de estabilidad a la señal.

IC2 es un conocido estabilizador de tensión de 5V/1A (7805CV), que es complementado por C5.

Q1 se encargan de suministrar la corriente necesaria para alimentar el rele, ya que el microcontrolador no tiene suficiente potencia para hacerlo directamente.

D2 absorbe los picos inversos de corriente que se producen al alimentar la bobina del rele, no se puede suprimir del esquema, ya que el transistor seria destruido con el tiempo.

C2 es el condensador de desacople del PIC. Este condensador es tan valioso como el mismo PIC así que no se puede suprimir y ha de estar lo mas cerca posible de los pines de alimentación del integrado.

Los LEDs solo se utilizan para saber en que estado se encuentra el circuito (a la hora de programar esto nos ayuda mucho para encontrar errores en el código del programa), se pueden suprimir ya que el circuito estará oculto.

El calculo de las resistencias (R8, R7) limitadoras de los diodos LED se hará de la siguiente forma:

R = (5V-1.2V) /0.015A,

Este es un cálculo genérico, siendo 5V la tensión de alimentación, 1.2V el voltaje de la unión del diodo LED (que varía según su color), y por ultimo 0.015A es el la corriente que deseamos atraviese el LED.

Para calcular la resistencia (R1) de polarización del transistor se utiliza una formula también genérica:

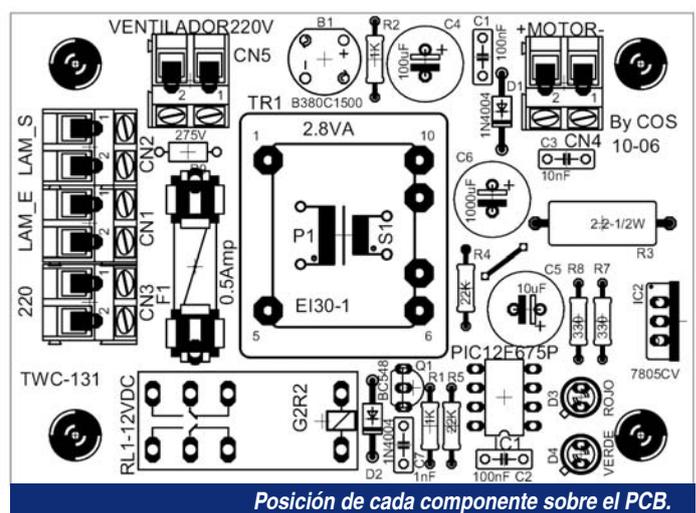
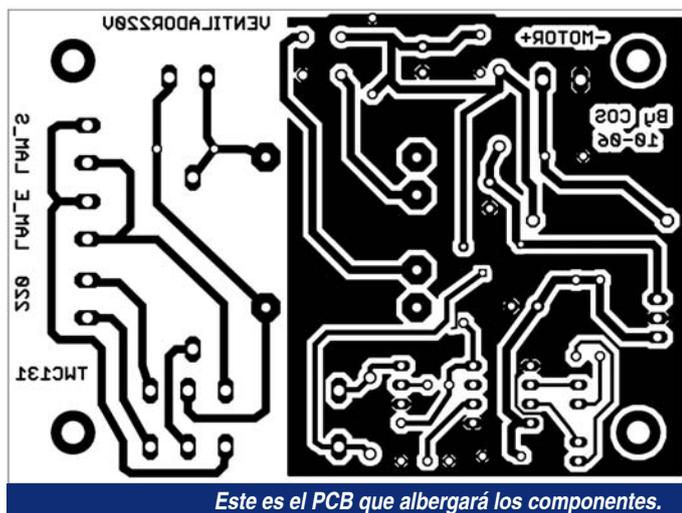
R1 = (5V-0.6V) / 0.001A,

Debemos recordar que el valor de 0.001A solo es aproximado. Para estar seguros que el transistor estará trabajando en saturación, el proceso es tan simple como multiplicar el valor de la corriente de polarización (en este caso 1mA) por la ganancia en corriente del transistor (presente en su hoja de datos). 5V es la alimentación, y los 0.6V corresponden a la caída de tensión entre los extremos de la unión base/emisor de un transistor de silicio.

La resistencia limitadora R3 de 1/2W en serie con el motor limita los picos de tensión durante el arranque. Es recomendable colocarla cuando se trabaja con motores y las fuentes de alimentación están demasiado justas de corriente. Se encarga de limitar la corriente del motor durante el arranque ya que esta es muy elevada con respecto a la corriente de trabajo indicada en el ventilador por el fabricante. ■

Este es el PCB que albergará los componentes.

Posición de cada componente sobre el PCB.

El programa con detección por entrada digital, totalmente comentado:

```
'****************************************************************************************
'NOMBRE: Temporizador_WC_13
'MICRO: PIC12F675
'DESCRIPCION: Automatismo para el control de un ventilador de WC, detección por entrada digital
'FECHA/AUTOR: By COS, 09/06, 10/06
'Version 1.0
'oscilador interno
'versión 1.1 soft, versión 1.1 hard, compatible versión 1.0 de soft con versión 1.1 de hard
'Cambio del soft para emplear una entrada que detecta perdida de alimentación en la lámpara
'simplificación del disparo del rele
'modificación del sistema de control de las salidas
'Versión 1.3
'se activa el WDT
'se cambia la rutina del control de flash de los led para hacerlos independientes uno del otro
'*****************************************************************************************
' *********************** Declaración de variables *****************
Dim timer_base As Word     'base patrón para los timer
Dim timer_base_aux As Word 'establece el tiempo en mSeg, en el que se basaran los timer
Dim contador As Byte       'variable que controla la fase en la que se encuentran las tempori-
zaciones
Dim timer1_sg As Word      'primera base de tiempos, para el control de rutinas
Dim timer1 As Word         'indica el tiempo para la base timer_sg
Dim led_flash_verde As Byte'permite que se ejecute la rutina que se
                           'encarga del parpadeo del led verde
Dim led_flash_rojo As Byte 'permite que se ejecute la rutina que se
                           'encarga del parpadeo del led rojo
Dim flash_verde As Byte    'indica dentro de la rutina si el próximo
                           'estado del led apagado o encendido, verde
Dim flash_rojo As Byte     'indica dentro de la rutina si el próximo
                           'estado del led apagado o encendido, rojo
Dim rele As Bit            'controla el estado del rele
Dim ledverde As Bit        'controla el estado del led verde
Dim ledrojo As Bit         'controla el estado del led rojo
' ************************ Asignación de valores de las variables *******
timer_base = 0             'inicializa el timer_base
timer_base_aux = 1000      'establece el desbordamiento de timer_base, 1Seg
contador = 0               'establece la fase cero del programa main
timer1_sg = 0              'inicializa el timer1_sg
timer1 = 30                'establece el desbordamiento de timer1_sg, 30Seg
led_flash_verde = 1        'flash del led verde habilitado
led_flash_rojo = 0         'flash del led rojo deshabilitado
flash_verde = 0            'estado inicial del flash del led verde
flash_rojo = 0             'estado inicial del flash del led rojo
rele = 0                   'estado inicial del rele, off
ledverde = 1               'estado inicial del led verde, on
ledrojo = 0                'estado inicial del led rojo, off
' *********************** Inicialización de registros generales y de E/S *********
VRCON = 0x00    'vref off (power off the comparator voltage)
ANSEL = 0x00    'off ADC
TRISIO = 0x00   'tri-state pins, are outputs
GPIO = 0x00     'clear port
GPIO.5 = 1      'pin a 1, maniobra invertida, activa a cero
GPIO.4 = 1      'pin a 1, maniobra invertida, activa a cero
GPIO.1 = 1      'para mantener la compatibilidad con la ver. 1.0 de hard
TRISIO.2 = 1    'GP2 pin, is input
CMCON = 0x07    'comparator off
WaitMs 10       'pausa de 10mSeg.
' *********** habilitación de interrupciones y programación del timer1 ****
INTCON.PEIE = 1            'bit de habilitación de interrupciones de periféricos
T1CON.TMR1ON = 1           'bit de habilitación del temporizador timer1
T1CON.TMR1CS = 0           'bit de selección de reloj para el timer1, interno Fosc/4
INTCON.T1IE = 1            'bit de habilitación de interrupción de TMR1 por rebose
T1CON.T1CKPS0 = 0          'bit de selección del prescaler para el reloj del timer1
T1CON.T1CKPS1 = 0          'bit de selección del prescaler para el reloj del timer1
```

```
TMR1H = 0xfc                'carga el byte alto del registro del tiemr1 (1mSeg)
TMR1L = 0x18                'carga el byte bajo del registro del timer1 (1mSeg)
PIE1.TMR1IE = 1             'activa la interrupción del timer1
OPTION_REG.TOCS = 0         'selecciona reloj interno para el WDT
OPTION_REG.PSA = 1          ' asigna el prescales al WDT
OPTION_REG.PS0 = 1          ' bit de la selección del factor de división para el WDT
OPTION_REG.PS1 = 1          ' bit de la selección del factor de división para el WDT
OPTION_REG.PS2 = 1          ' bit de la selección del factor de división para el WDT
INTCON.TOIE = 0             ' deshabilito interrupción por el trm0
Enable                      'INTCON.GIE=1, habilita las interrupciones generales


' ************* Rutina del programa ***************************************
main:
      ASM:      clrwdt                    'reinicializa el WDT antes que se desborde
      If contador = 0 Then                'primera fase del control de tiempos y maniobras
            If timer1_sg >= timer1 Then   'si pasan los primeros 30Seg.
                  contador = 1            'permite la siguiente fase
                  timer1 = 0              'configura la base de tiempos con un nuevo valor 0Seg
                  timer1_sg = 0           'activa la base de tiempos
                  led_flash_verde = 0     'desconecta el parpadeo del led verde
                  led_flash_rojo = 1      'conecta el parpadeo del led rojo
                  ledverde = 0            'apaga el led verde
                  ledrojo = 1             'prende el led rojo
                  rele = 0                'desconecta el rele
            Endif
      Endif
      If contador = 1 Then                'segunda fase del control de tiempos y maniobras
            If timer1_sg >= timer1 Then   'cuando termina la base de tiempos
                                          'según el ultimo valor del timer1
                  If GPIO.2 = 0 Then      'comprueba que hay alimentación en la fuente
                        contador = 2      'permite la siguiente fase
                        timer1 = 120      'carga la base de tiempos para que cuente 2minutos
                        timer1_sg = 0     'activa la base de tiempos
                        flash_verde = 1   'sincroniza el destello con el led rojo
                        flash_rojo = 1    'sincroniza el  destello con el led verde
                        led_flash_verde = 1 'permite el parpadeo del led verde
                        led_flash_rojo = 1  'permite el parpadeo del led rojo
                        ledverde = 1      'prende el led verde
                        ledrojo = 1       'prende el led rojo
                        rele = 1          'energiza la bobina del rele
                  Endif
            Endif
      Endif
      If contador = 2 Then                'tercera fase del control de tiempos y maniobras
            If timer1_sg >= timer1 Then   'cuando pasan los 120seg.
                  contador = 1            'se cambia de fase
                  timer1 = 1              'se prepara la base de tiempos timer1_sg, para
                                           que cuente 1Seg.
                  timer1_sg = 0           'activa el contaje de la base de tiempos ti
                                           mer1_sg
                  rele = 0                'desconecta el rele
                  flash_verde = 1         'encendido alternativo del led verde con res
                                           pecto al rojo
                  flash_rojo = 0          'encendido alternativo del led rojo con res
                                           pecto al verde
                  led_flash_verde = 1     'permite el parpadeo del led verde
                  led_flash_rojo = 1      'permite el parpadeo del led rojo
                  ledverde = 1            'activa el led verde
            ledrojo = 1                   'activa el led rojo
            Endif
      Endif
Goto main
End
```

```
 On Interrupt  'Comienzan las rutinas de las interrupciones, desactiva las interrupciones
       Save System  'Guarda los valores del sistema

'-------------------------------------- control salidas -------------------------------------

       If ledverde = 1 Then       'controla el estado de la salida del led verde
              GPIO.5 = 0          'prende led verde
       Else
              GPIO.5 = 1          'apaga led verde
       Endif
       If ledrojo = 1 Then        'controla la salida del led rojo
              GPIO.4 = 0          'prende led rojo
       Else
              GPIO.4 = 1          'apaga led rojo
       Endif
       If rele = 1 Then           'controla la salida del rele
              GPIO.0 = 1          'energiza rele
              GPIO.1 = 0          'energiza rele, compatibilidad con la ver. 1.0 de hard
       Else
              GPIO.0 = 0          'desconecta el rele
              GPIO.1 = 1          'desconecta el rele, compatibilidad con la ver. 1.0 de hard
       Endif

'------------------------------ bases de tiempos y flash diodos -----------------------------
       If PIR1.TMR1IF = 1 Then    'comprueba que es esta la interrupción activa
                           '(por costumbre, en este caso solo hay una)
              timer_base = timer_base + 1             'se incrementa con cada desbordamiento
                                                       del timer1
              If timer_base >= timer_base_aux Then    'control del numero de desbordamientos
                                                'según el valor de timer_base_aux
                     If timer1_sg < timer1 Then timer1_sg = timer1_sg + 1  'base de tiempos
                                                                             timer1
                     If led_flash_verde > 0 Then      'se encarga de hacer el led verde in
                                                       termitente
                            If flash_verde = 0 Then   'controla el parpadeo del led verde
                                   ledverde = 0       'conecta el verde
                                   flash_verde = 1    'variable de control, permite que se
                                                       conecte el verde
                            Else
                                   ledverde = 1       'desconecta el verde
                                   flash_verde = 0    'variable de control, controla el parpadeo
                            Endif
                     Endif
                     If led_flash_rojo > 0 Then       'se encarga de hacer el led rojo inter
                                                       mitente
                            If flash_rojo = 0 Then    'controla el parpadeo del rojo
                                   ledrojo = 0        'conecta el led rojo
                                   flash_rojo = 1     'variable de control, permite que se
                                                       conecten los led
                            Else
                                   ledrojo = 1        'desconecta el led rojo
                                   flash_rojo = 0     'variable de control, controla el parpadeo
                            Endif

                     Endif
                     timer_base = 0      'se reinicialaza el valor de la base de tiempos patrón
              Endif
       Endif
       TMR1H = 0xfc          'carga el registro del timer1, para que desborde cada 1mSeg.
       TMR1L = 0x18
       PIR1.TMR1IF = 0       'borra el flag de salto del tmr1
Resume              '      activa las interrupciones y retorna al programa
```

Nota: El programa ha sido escrito para el Basic del Pic Simulator IDE, se puede obtener una versión gratuita funcional por tiempo limitado desde http://www.oshonsoft.com/

usando LCDs primera parte

Los displays LCD permiten que nuestros proyectos tengan una presentación óptima, a la vez que le proporcionan mayores funcionalidades al operador. En esta nota veremos cuales son sus características principales y aprenderemos como utilizarlos.

En la mayoría de los dispositivos electrónicos modernos se necesita visualizar valores, modificar parámetros, representar estados o barras de avance, etcétera.

Si bien existen desde hace mucho tiempo los display de siete segmentos, en sus diferentes versiones, en aplicaciones donde sea necesario mostrar valores que exceden los 4 dígitos, ya conviene utilizar un display LCD.

La razón primaria al realizar esta elección es simple, utilizar displays de segmentos exige en la mayoría de los casos utilizar técnicas de multiplexación en el microcontrolador, de forma de reducir líneas de entrada / salida aplicadas al manejo de los displays porque de otra forma estos se necesitarian de 8 pines de salida por cada digito.

Las técnicas de multiplexación nos permiten manejarlos con menos pines, pero como nada es gratis, se pierden numerosos recursos de memoria para manejar la multiplexación dentro del micro, además de "robar tiempo" al programa principal. A veces, con programas complejos, se torna difícil mantener bien compensados los tiempos, y ni hablar de los problemas que pueden aparecer cuando hay que manejar interrupciones de otros periféricos.

Todas estas razones hacen que el display LCD, que además consume menos energía que un sistema de displays de segmentos, se torne una excelente opción al momento de elegir el modo de mostrar información en un proyecto.

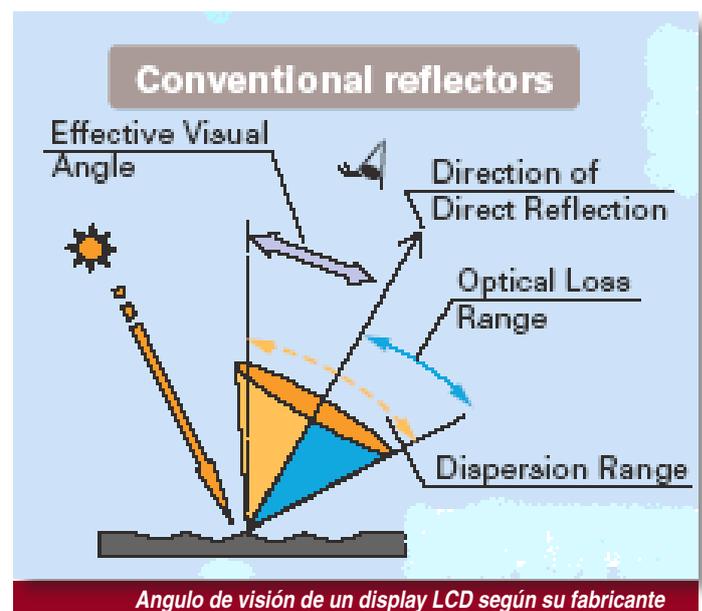
.Como funcionan los displays LCD

El funcionamiento de estas pantallas se fundamenta en sustancias que comparten las propiedades de sólidos y líquidos a la vez. Cuando un rayo de luz atraviesa una partícula de estos materiales tiene necesariamente que atravesar el espacio vacío que hay entre sus moléculas, como lo haría atravesar un cristal sólido, pero a cada una de estas partículas se le puede aplicar una corriente eléctrica para que cambie su polarización dejando pasar a la luz (o no).

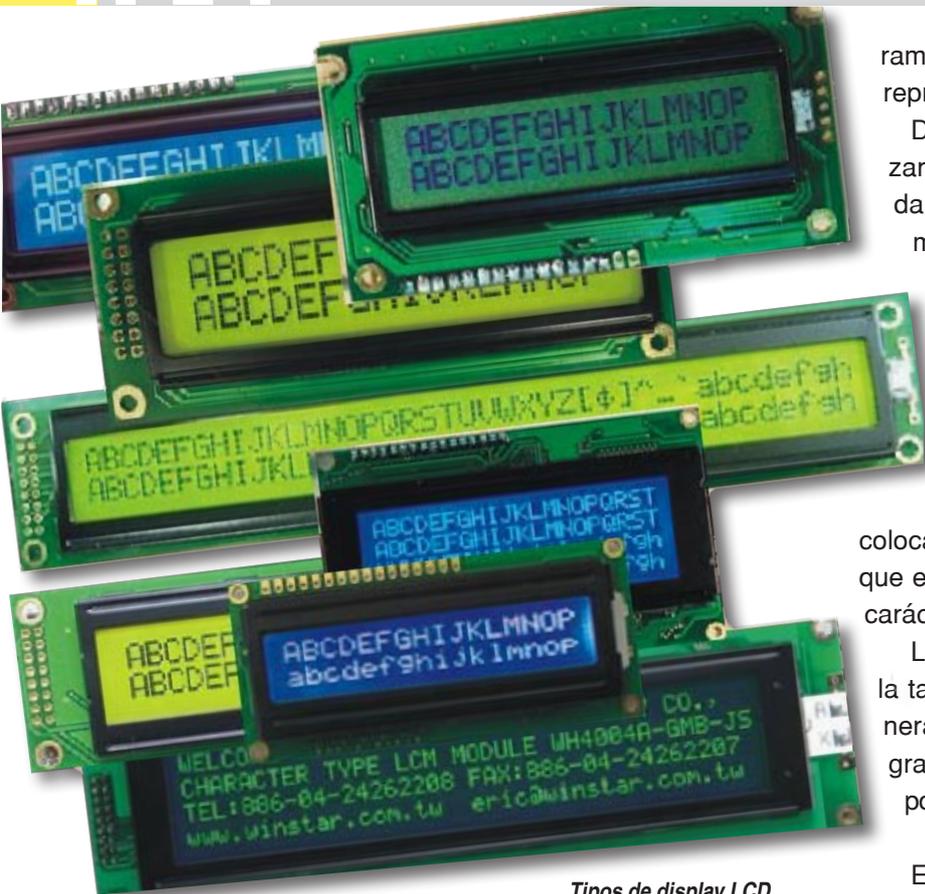
Una pantalla LCD esta formada por 2 filtros polarizados, colocados perpendicularmente, de manera que al aplicar una corriente eléctrica al segundo de ellos dejaremos pasar o no la luz que ha atravesado el primero.

Normalmente la visión del carácter representado en el display se produce por refracción de la luz en el mismo, y tiene que ver con el ángulo desde donde se lo mire, ya que variándolo se verá con mayor o menor claridad o definición.

Conventional reflectors
Effective Visual Angle
Direction of Direct Reflection
Optical Loss Range
Dispersion Range

Angulo de visión de un display LCD según su fabricante

.Tipos de displays LCD

Los displays LCD que trataremos en esta nota son del tipo alfanumérico, en nuestro caso nos referiremos al modelo de 2 líneas y 16 caracteres por línea. Es uno de los más utilizados del mercado.
También se fabrican displays de 1, 2, 3 y 4 líneas por 8 caracteres, 16 caracteres, 20 caracteres y también 40 caracteres por línea.

Por cada modelo varían los colores del display y pueden disponer de retroiluminador incorporado o no, dependiendo esta elección del lugar donde se instale el dispositivo que lo contenga y si hay luz natural o no.

Tipos de display LCD del mercado

Otra de las diferencias que pueden tener entre ellos es la forma de la matriz de puntos con la cual conforman los caracteres. Existen de 5 columnas por 7 filas (el modelo al cual nos referiremos durante la nota usa este tipo) y de 5 columnas por 8 filas. Este último puede utilizar el cursor en la última fila de cada caracter.

Otra caracteristica que diferencian a estos dispositivos es el controlador que utilizan. Citaremos solamente el HD44780 de la firma Hitachi, ya que se trata de uno de los más difundidos.

Es muy importante al momento de seleccionar un display LCD, tener en cuenta qué tabla de caracteres utiliza. Esto se determina normalmente en su hoja de datos.

Este controlador dispone de una tabla de caracteres ya grabados en su memoria ROM, que el fabricante pone en su hoja de datos.

Cabe aclarar a esta altura, que es muy importante al momento de seleccionar un display LCD, tener en cuenta que tabla de caracteres utiliza. Esto se determina normalmente en su hoja de datos, donde el fabricante debe colocar dicha información.

En caso de no tener en cuenta este detalle, seguramente nos encontraremos en problemas al querer representar un carácter que no existe en su tabla.

De todos modos para estos casos se puede utilizar la memoria de generación de caracteres, llamada CGRAM en la jerga de los displays LCD. Esta memoria nos permite generar solo 8 caracteres definidos por el usuario, así que no se puede abusar de la misma.

Además la generación de estos caracteres se debe realizar cada vez que se inicia el display LCD, cargándolos uno por uno en las posiciones de memoria CGRAM, para luego al utilizarlos el display los tomara de allí y los colocara en la memoria de display, llamada DDRAM, que es la misma que se escribe cuando enviamos un carácter para su visualización.

La única diferenciación que existe entre el uso de la tabla de caracteres de ROM y los caracteres generados en CGRAM es que estos últimos no quedan grabados en el display, ya que es memoria volátil, por lo tanto deberán ser cargados cada vez que se inicializa el display.

En cuanto a su uso, para el display es lo mismo tomar el carácter desde ROM que desde CGRAM.

Tabla de caracteres del HD44780

Lower 4bit \ Higher 4bit	0000	0010	0011	0100	0101	0110	0111	1010	1011	1100	1101	1110	1111
xxxx0000			0	@	P	`	p		ー	タ	ミ	α	p
xxxx0001		!	1	A	Q	a	q	。	ア	チ	ム	ä	q
xxxx0010		"	2	B	R	b	r	「	イ	ツ	メ	β	θ
xxxx0011		#	3	C	S	c	s	」	ウ	テ	モ	ε	∞
xxxx0100		$	4	D	T	d	t	、	エ	ト	ヤ	μ	Ω
xxxx0101		%	5	E	U	e	u	・	オ	ナ	ユ	σ	ü
xxxx0110		&	6	F	V	f	v	ヲ	カ	ニ	ヨ	ρ	Σ
xxxx0111		'	7	G	W	g	w	ァ	キ	ヌ	ラ	g	π
xxxx1000		(	8	H	X	h	x	ィ	ク	ネ	リ	√	x̄
xxxx1001		)	9	I	Y	i	y	ゥ	ケ	ノ	ル	⁻¹	y
xxxx1010		*	:	J	Z	j	z	ェ	コ	ハ	レ	j	千
xxxx1011		+	;	K	[	k	{	ォ	サ	ヒ	ロ	ˣ	万
xxxx1100		,	<	L	¥	l	|	ャ	シ	フ	ワ	¢	円
xxxx1101		-	=	M	]	m	}	ュ	ス	ヘ	ン	£	÷
xxxx1110		.	>	N	^	n	→	ョ	セ	ホ	゛	ñ	
xxxx1111		/	?	O	_	o	←	ッ	ソ	マ	゜	ö	█

Distribución de los pines en distintos modelos de display

.Pines de conexión

En el conector del display LCD nos encontraremos con 14 pines, dispuestos en dos líneas a la izquierda del display, o una línea en la parte superior o inferior del display.

.Función de los pines de conexión

Los pines de conexión de los displays LCD incluyen un bus de datos completo de 8 bits, un pin de habilitación E (de Enable), un pin de selección llamado RS (Register Select) que según su estado indicara si es una instrucción o si se trata de un carácter a escribir y por ultimo un pin que indica si se va a leer o escribir en el display, este ultimo pin se llama R/W (Read/Write).

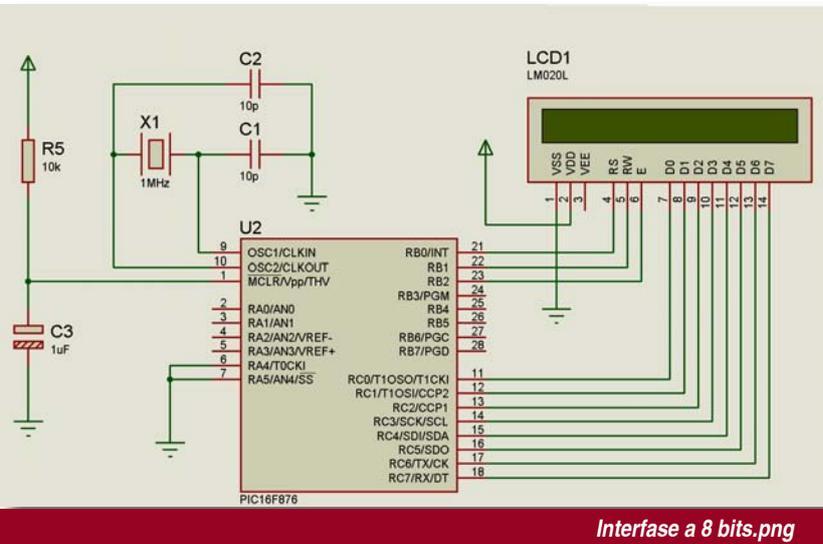
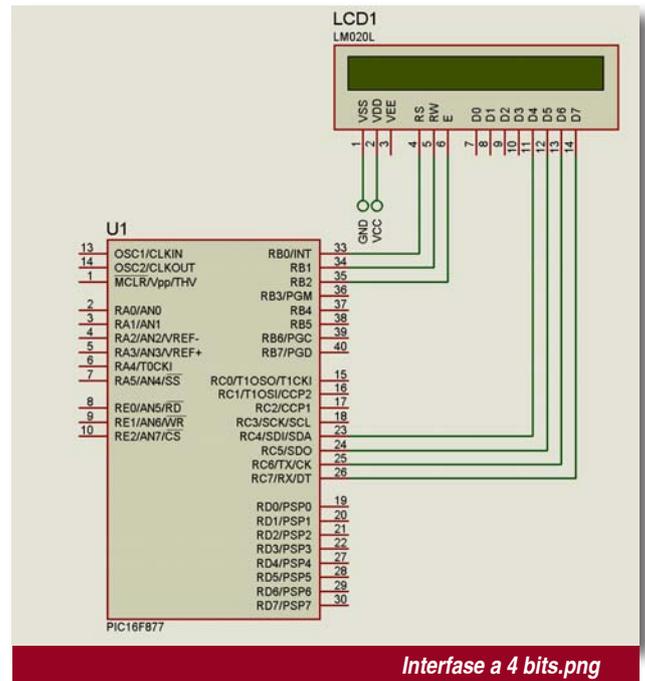
Por supuesto además están los pines de alimentación Vss y Vdd, un pin de regulación del contraste llamado Vo y algunos modelos de display, con retroiluminador incorporado, pueden incluir dos pines adicionales para la alimentación del mismo, llamados A y K (por Anodo y Katodo).

En la hoja de datos del modulo LCD que utilicemos el fabricante nos informara el diagrama de tiempos y estados de las señales, para establecer la comunicación con el modulo LCD, tanto para enviar ordenes de configuración o datos a escribir en el display, estos tiempos deben ser respetados si se quiere llegar a buen puerto con el manejo eficaz del mismo.

Función de los pines

Control y Datos / INSTRUCCIONES	Señal de control		DATO / DIRECCION							
	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0
Borrar la pantalla	0	0	0	0	0	0	0	0	0	1
Cursor a casa (posicion 0)	0	0	0	0	0	0	0	0	1	x
Seleccionar Modo	0	0	0	0	0	0	0	1	ID	S
Enciende/apaga pantalla	0	0	0	0	0	0	1	D2	C	B
Desplazar Cursor / Pantalla	0	0	0	0	0	1	SC	RL	x	x
Activar Funcion	0	0	0	0	1	DL	1	0	x	x
CG RAM (generador caract.)	0	0	0	1	Direccion generador RAM					
DD RAM (memoria caract.)	0	0	1	Direcciones de datos RAM						
Bandera de ocupado	0	1	0	AC						
Escribir CG RAM / DD RAM	1	1	Escritura de Datos							
Lectura CG RAM / DD RAM	1	1	Lectura de Datos							

.Interfaces con microcontroladores

El display LCD puede ser conectado a un microcontrolador utilizando 4 pines de datos u 8 pines de datos, como vemos en las imágenes.

Interfase a 8 bits.png

Interfase a 4 bits.png

.Conjunto de instrucciones

El controlador HD44780 responde a un conjunto especial de instrucciones que le permite configurarlo y utilizarlo.

Entre las funciones permitidas están las siguientes:

-Utilizarlo en modo a 4 u 8 bits

-Borrar la pantalla

-Mover el cursor o mover el texto dejando el cursor fijo

-Hacer parpadear el carácter donde esta posicionado el cursor

-Mostrar o esconder el cursor

-Desplazar el texto por la pantalla

-Encender y apagar el display

En la siguiente tabla veremos como serán los valores a enviar y los estados de los pines de control para hacerlo.

Control y Datos / INSTRUCCIONES	Señal de control		DATO / DIRECCIÓN								
	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	DESCRIPCIÓN DE LAS OPERACIONES
Borrar la pantalla	0	0	0	0	0	0	0	0	0	1	Borra el display completo y pone en 0 el contador de direcciones DDRAM
Cursor a casa (posición 0)	0	0	0	0	0	0	0	0	1	--	Pone en 0 el contador de direcciones DDRAM. Vuelve a 0 si se estaba desplazando el display
Seleccionar Modo	0	0	0	0	0	0	0	1	I/D	S	Elije dirección de desplazamiento del cursor y especifica si desplaza el texto
Enciende/apaga pantalla	0	0	0	0	0	0	1	D	C	B	Enciende/apaga el display (D). Muestra o esconde el cursor (C). Parpadeo del carácter que señala el cursor (B)
Desplazar Cursor / Pantalla	0	0	0	0	0	1	S/C	R/L	--	--	Mueve el cursor y desplaza el display sin cambiar el contenido de la memoria DDRAM
Activar Función	0	0	0	0	1	DL	N	F	--	--	Elige entre interfase de 4 u 8 bits. Elige numero de líneas del display y fuente de los caracteres a mostrar.
CG RAM (generador caract.)	0	0	0	1	dirección generador RAM						Elige la dirección de CGRAM. Una vez elegida se puede leer o escribir sobre esa dirección.
DD RAM (memoria caract.)	0	0	1	Direcciones de datos RAM							Elige la dirección de DDRAM. Una vez elegida se puede leer o escribir sobre esa dirección.
Bandera de ocupado	0	1	0	AC							Lectura de la bandera de display ocupado, para determinar en que momento se puede acceder.
Escribir CG RAM / DD RAM	1	1	Escritura de Datos								Escritura de datos en DDRAM o CGRAM
Lectura CG RAM / DD RAM	1	1	Lectura de Datos								Lectura de datos desde DDRAM o CGRAM

I/D = 1: Incrementa
I/D = 0: Decrementa
S = 1: Elige desplazamiento del texto
S/C = 1: Desplaza el Display
S/C = 0: Mueve el cursor
R/L = 1: Desplaza a la derecha
R/L = 0: Desplaza a la izquierda
DL = 1: Interfase a 8 bits

DL = 0: Interfase a 4 bits
N = 1: Activa las 2 líneas
N = 0: Activa solo 1 línea
F = 1: 5 * 10 puntos
F = 0: 5 * 8 puntos
BF = 1: Operando internamente
BF = 0: Acepta instrucciones

Tabla de operaciones del display

.Secuencia de inicialización

El display debe ser inicializado para poder ser utilizado, además debe ser enviado un comando de borrar pantalla una vez inicializado (aunque esto no esta documentado en la mayoría de las hojas de datos de los displays) para poder utilizarlo sin inconvenientes.

Según la interfase que vaya a utilizarse, si es una interfase a 4 u 8 bits esta secuencia es diferente.

Vemos en las figuras siguientes los pasos a seguir para inicializar el display para cada una de las interfases.

Vemos que lo único que cambia en la interfase a 4 bits es que una vez que esta enviado el comando de inicialización, a partir de allí todo envío debe ser realizado enviando primero el nibble alto y luego el nibble bajo de cada byte que estemos enviando.
Esto también incluye a los comandos de inicialización restantes luego de definir la interfase.

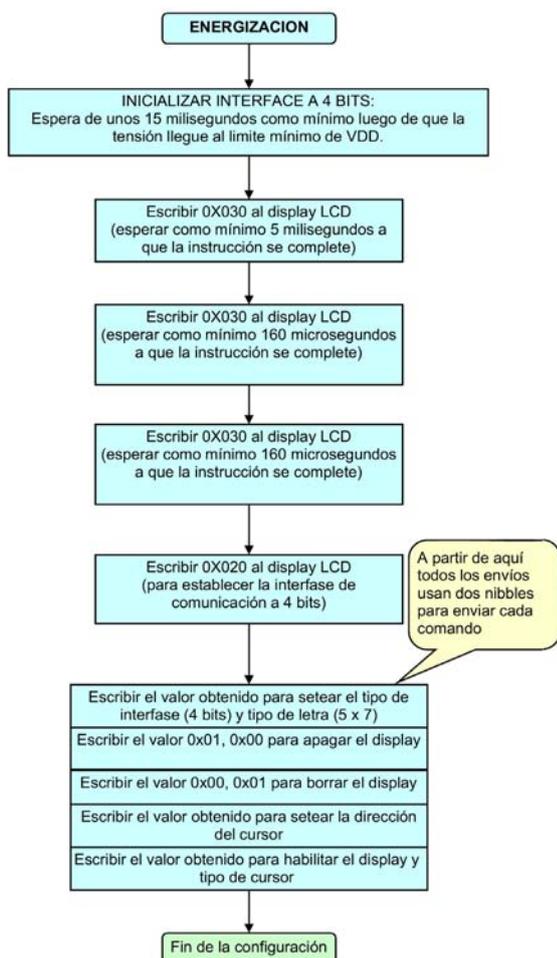
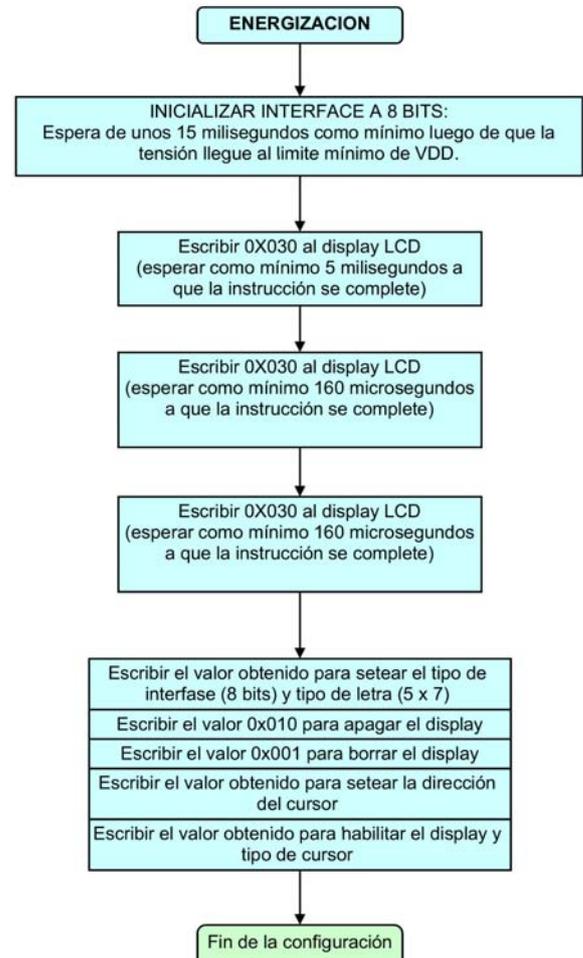
Como corolario a esta nota (la primera de una serie sobre displays LCD), cabe acotar que en la mayor parte de los proyectos que se realizan no es necesario utilizar la línea R/W, por lo cual se pone esta línea a nivel bajo (conectada a GND) y se respetan los tiempos necesarios y casi la totalidad de los displays LCD funcionan correctamente de este modo.

Sin embargo, los fabricantes de displays LCD aconsejan la utilización de la línea R/W conectada al controlador, cada vez que se escribe sobre el display, antes de enviar un nuevo comando, se pasan las líneas de datos a entradas y se lee el estado del BIT 7 del byte entregado por el display, de este modo sabremos cuando el display esta listo para recibir un nuevo comando.

Si bien parece una tontería, en aplicaciones complejas, donde los tiempos valen oro, es muy importante no desperdiciarlo, por lo tanto utilizando la línea R/W se puede optimizar mucho el aprovechamiento de los mismos.

Imaginen además la ventaja de adaptarse sin problemas a distintas marcas y modelos de displays, funcionando a distintas condiciones de temperaturas y humedades, sin cambiar una sola línea de código.

En la próxima nota, hablaremos del manejo de estos dispositivos en tres de los lenguajes mas utilizados en programación de PICs. Lo haremos en Assembler, ***PicBasic y C de CCS.***

Referencias:
Wikipedia, la enciclopedia libre.
Winstar, fabricante de displays LCD.
Hoja de datos del controlador HD44780, de la firma Hitachi.

Secuencia inicialización para interfase a 4 bits

Secuencia inicialización para interfase a 8 bits

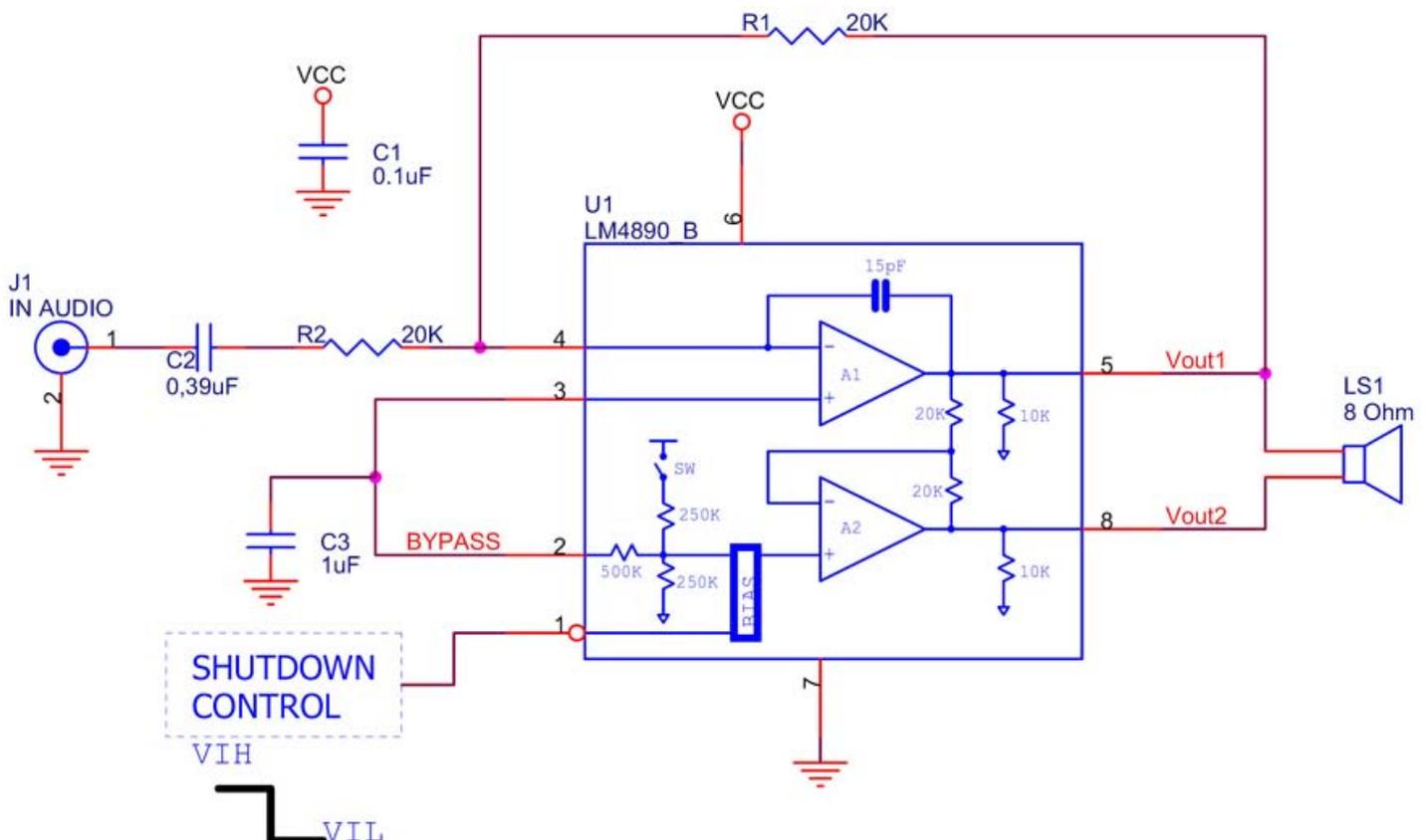
los herederos del LM386

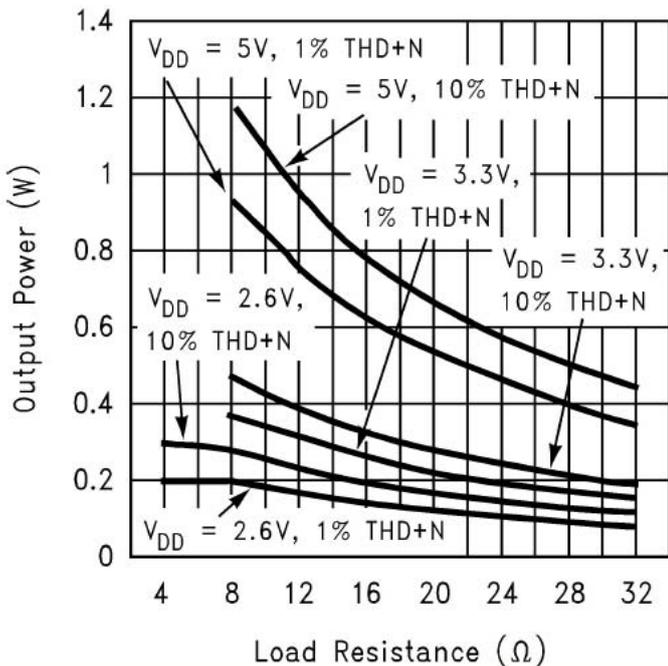
Cuando hablábamos de un amplificador versátil, pequeño, de potencia aceptable para aplicaciones móviles, de pocos componentes externos, económico y con muchas ventajas más, era un automatismo decir: LM386.
De la mano de National Semiconductor, esto está empezando a cambiar.

El LM386 siempre fué una excelente elección para los diseñadores al momento de utilizar un pequeño amplificador de audio (1W).

Uno de los aspectos que podía llegar a complicar dicha selección era el tamaño y costo que significaban los pocos, pero necesarios, capacitores electrolíticos que éste CI requería para su funcionamiento.

National Semiconductor, está comenzando a introducir pequeños CI en encapsulado SMD los que, son capaces de entregar una potencia media continua de un 1W sobre una carga de 8 Ω, con una distorsión inferior al 1% a partir de una tensión de alimentación de 5 V. Un claro ejemplo de esto, es el CI LM4890, un amplificador de potencia de audio diseñado sobre todo, para usos exigentes en teléfonos móviles y otros usos portables tales cómo *Notebooks* y *Palms*.

Los amplificadores de audio de Tecnología Boomer, fueron diseñados específicamente para proveer salida de alta calidad y potencia, con una cantidad mínima de componentes externos. Gracias a la configuración puente (BTL), el LM4890 no requiere en la conexión de salida, los clásicos capacitores electrolíticos de gran tamaño. Esta propiedad convierte al LM4890 en el componente ideal para el uso en telefonía móvil y en aplicaciones de baja tensión, donde un requisito esencial es el consumo mínimo de energía.

Circuito de aplicación, clásico para el LM4890

Output Power vs Load Resistance

Nótese en las curvas, la posibilidad de operación a partir de 2,5 V.

Seguramente el LM386 seguirá siendo el amplificador preferido por muchos para sus aplicaciones tradicionales

Ofrece además, un modo de operación en baja potencia o Stand-By, que es activado a través de un estado lógico alto, aplicado en el pin 1 (para encapsulados MSOP y SO). Consumiendo en éste modo de operación, la irrisoria cantidad de 0.1 μA.

Este CI permite configurar su ganancia a través de un par de resistores externos, posee protección térmica interna, supresor de los clásicos ruidos de encendido y apagado.

El camino del cambio ha comenzado con esta línea de amplificadores de National Semiconductor. Seguramente el LM386 seguirá siendo el amplificador preferido por muchos para sus aplicaciones tradicionales, cómo en otros rubros lo son el Timer 555, el OA 741 y el PCI 16F84A, pero nada se crea ni se pierde, sino que se transforma.

Para más información acerca de éste modelo de amplificadores de audio, pueden visitar el sitio oficial de National Semiconductor: http://www.national.com/mpf/LM/LM4890.html.

PIC BASIC

La revista uControl contará con varios tutoriales que se irán entregando en capítulos, uno por número. En este caso, te enseñaremos a programar microcontroladores en BASIC, usando para ello el compilador PIC BASIC incluido en el PIC SIMULATOR IDE (PSI). En esta primera entrega, veremos los aspectos más importantes del lenguaje.

PSI es una excelente herramienta, de muy bajo costo, que permite programar en BASIC y además simular el comportamiento de nuestro programa. Dispone para ello de gran variedad de módulos (LCD, teclados, LEDs, etc) que pueden conectarse de manera virtual a nuestro microcontrolador. Seguramente incluiremos una serie de artículos sobre él en los próximos números.

Comencemos a ver los elementos indispensables para crear un programa en lenguaje BASIC.

.Variables:

La programación sería prácticamente imposible sin el uso de variables. Podemos hacernos una imagen mental de las variables imaginándolas como una caja en la que podemos guardar algo. Esa caja es una de las muchas de que disponemos, tiene en el frente pegada una etiqueta con su nombre y ciertas particularidades, que hace que solo se puedan guardar en ella determinados tipos de objetos.

En esta analogía, cada caja es una variable, su contenido es el valor que adopta, y la etiqueta es el nombre de la variable. Como su nombre lo indica, y como veremos más adelante, el contenido de una variable puede ser modificado a lo largo del programa.

La programación sería prácticamente imposible sin el uso de variables.

.Tipos

En BASIC tenemos distintos tipos de variables, según el dato que puedan almacenar:

Bit (un bit de longitud, almacena 0 o 1 únicamente)

Byte (un byte de longitud, almacena números enteros entre 0 y 255)

Word (dos bytes de longitud, almacena números enteros entre 0 y 65,535)

Long (cuatro bytes de longitud, almacena números enteros entre 0 y 4,294,967,295)

El tipo "Long" solo está disponible mediante un modulo opcional al PIC SIMULATOR IDE. A diferencia de otros BASIC, la declaración de variables puede ser hecha en cualquier parte del programa, y todas son consideradas globales, es decir, su valor es accesible desde todas las subrutinas y zonas del programa. El numero de variables esta lógicamente limitado al monto de memoria RAM disponible en cada microcontrolador.

.DIM

Las variables deben ser declaradas antes de utilizarlas, mediante la instrucción DIM, como se muestra en los siguientes ejemplos:

```
DIM A AS BIT
DIM B AS BYTE
DIM X AS WORD
DIM Y AS LONG
```

También es posible utilizar vectores, que son una matriz de dimensiones 1xN. Por ejemplo, la sentencia siguiente:

```
DIM A(10) AS BYTE
```

La instrucción anterior declara un vector (al que

nos referiremos algunas veces como “array”) de diez elementos del tipo BYTE, que serán accedidos mediante el uso de subíndice (entre paréntesis) del 0 al 9.

Las variables tipo Word, como vimos, están compuestas por dos bytes. El primero de ellos es llamado byte “alto” y el otro “bajo”, dado que el primero contiene los 8 bits más significativos. En BASIC podemos acceder individualmente a cada uno de los bytes que componen un Word mediante las extensiones “.HB” (High byte, o byte alto) y “.LB” (Low Byte o byte bajo). Veamos un ejemplo:

```
DIM A AS BYTE
DIM B AS WORD
A = B.HB
A = B.LB ‘Esto es lo mismo que A = B
B.HB = A
B.LB = A
B = A ‘Esto también borra el byte alto de
la variable B
```

Los bits individuales de cada variable pueden ser accedidos individualmente también, simplemente poniendo como extensión “.n” donde “n” es el numero de bit (1,2, 3, etc.)

```
DIM A AS BYTE
DIM B AS BIT
B = A.1
B = A.7
A.0 = A.5
```

.RESERVE

La sentencia RESERVE le permite al programador reservar un numero de posiciones de la RAM para su uso en rutinas en lenguaje ensamblador o para el In-Circuit Debugger de MPLAB. Simplemente, si queremos reservar 20 bytes de RAM, escribimos:

```
RESERVE 20
```

.Puertos

Todos los registros del microcontrolador están disponibles para usar en los programas BASIC, como si se tratase de variables del tipo BYTE con el nombre del registro utilizado en las datasheet (PORTA, PORTB, TRISA, etc.). Por supuesto, se puede acceder a bits individuales de los registros con la técnica vista párrafos atrás. Algunos ejemplos:

```
TRISA.1 = 0
TRISB = 0
PORTA.1 = 1
PORTB = 255
STATUS.RP0 = 1
INTCON.INTF = 0
```

Existe una “forma corta” de acceder a los bits individuales de cada puerto, simplemente usando las variables BASIC tipo byte RA, RB, RC, RD, RE o bien las tipo bit RA0, RA1, RA2, ..., RE6, RE7

```
RA = 0xFF
RB0 = 1
```

.Punteros

En BASIC también podemos usar punteros. En realidad, cualquier variable definida como tipo BYTE o WORD pude ser usada como un puntero de memoria, usándola como argumento de la función POINTER. El valor contenido por la variable debe tener un valor comprendido entre 0 y 511.

Ejemplos:

```
DIM X AS WORD
DIM Y AS BYTE
X = 0x3F
Y = POINTER(X)
Y = Y + 0x55
X = X - 1
POINTER(X) = Y
Y = 0xAA
X = X - 1
POINTER(X) = Y
```

.SYMBOL

Una forma de escribir programas que nos resulten mucho más fáciles de entender es el uso de nombres simbólicos, o SYMBOL. Un “symbol” es una cadena que contiene código, asignado a un nombre. Al momento de compilar, PIC BASIC hace la “búsqueda y reemplazo” de nuestros símbolos y luego genera el código ASM y el HEX. Supongamos que tenemos un LED conectado al bit cero del puerto B. Mediante SYMBOL podemos hacer:

```
SYMBOL LED1 = PORTB.0
```

Luego, si queremos encender el LED, en lugar de

```
PORTB.0 = 1
```

podemos hacer

```
LED1 = 1
```

que es mucho más claro y fácil de leer. El código que aparece a la derecha del igual no puede contener instrucciones o comandos.

Las constantes (valores que usamos en nuestro programa, y que, por ejemplo, asignamos a las variables) pueden ser escritas en decimal (directamente el valor), en hexadecimal (anteponiendo “0x” o posponiendo “H” al valor) o en binario (anteponiendo “%” al valor). Por ejemplo:

```
DIM A AS BIT
DIM B AS BYTE
A = TRUE
B = 0x55
B = %01010101
```

Por supuesto, se pueden asignar nombres a las constantes, usando la instrucción CONST:

```
DIM A AS WORD
CONST PI = 314
A = PI
```

Hay tres instrucciones para el manejo individual de bits, que si bien no hacen nada que no se puede resolver con otras instrucciones o símbolos, ayudan mucho en la lectura del código. Se tratan de HIGH, LOW y TOGGLE, que ponen el bit en alto, bajo o lo invierten, respectivamente. Importante: Si el bit implicado como argumento de una de estas instrucciones es un bit de un puerto, el mismo bit en el TRIS correspondiente es puesto en cero, y dicho pin queda configurado como salida. Algunos ejemplos:

```
HIGH PORTB.0
LOW ADCON0.ADON
TOGGLE OPTION_REG.INTEDG
```

.GOTO

Esta es una de las instrucciones más polémicas que se encuentra en todos los dialectos BASIC. GOTO significa literalmente “IR A”, y sirve justamente para eso: desviar el flujo del programa a otro punto.

Para usar GOTO, es necesario poner una etiqueta en el lugar al que queremos “saltar”. Las etiquetas son simplemente nombres terminados en “:”, tal como se ve a continuación:

```
...
...
calculos:
...
...
...
...
...
GOTO calculos
...
...
```

En el ejemplo anterior, el programa se ejecutará hasta encontrar la instrucción “GOTO calculos”, que hará que se ejecuten nuevamente las instrucciones siguientes a la etiqueta “calculos:”. Cabe aclarar que las etiquetas no son un código ejecutable, es decir, no realizan ninguna acción, solo son un “marcador” del lugar al que se puede saltar con GOTO.

.Operaciones Lógicas y Matemáticas

PIC SIMULATOR IDE dispone de cinco operaciones matemáticas básicas, disponibles para las variables tipo Byte y Word. Estas son la suma (operador +), la sustracción (operador -), el producto (operador *), el cociente (operador /) y el módulo (operador MOD) .Por supuesto, el compilador es capaz de combinarlas para obtener operaciones matemáticas más complejas.

```
DIM A AS WORD
DIM B AS WORD
DIM X AS WORD
A = 123
B = A * 234
X = 2
X = (12345 - B * X) / (A + B)
```

Es posible calcular raíces cuadradas (aunque el resultado debe ser entero) con la función SQR:

```
DIM A AS WORD
A = 3600
A = SQR(A)
```

Para las variables de tipo Bit existen siete operaciones lógicas disponibles. Solo es posible efectuar una operación lógica por instrucción; aunque es muy posible que próximas versiones permitan más flexibilidad. Esté al tanto de las novedades! Estas operaciones también están disponibles para variables tipo Word o Byte. Veamos algunos ejemplos:

```
DIM A AS BIT
DIM B AS BIT
DIM X AS BIT
X = NOT A
X = A AND B
X = A OR B
X = A XOR B
X = A NAND B
```

```
X = A NOR B
X = A NXOR B
DIM A AS WORD
DIM B AS WORD
A = A OR B
PORTB = PORTC AND %11110000
```

.Mi primer programa: *Un LED parpadeando*

Luego de toda esta introducción puramente teórica, estamos en condiciones de encarar nuestro primer programa. A diferencia de un programa de ordenador, donde uno escribe el programa, lo compila, lo ejecuta y ya, en el mundo de los microcontroladores hay que, previamente, definir el tipo de microcontrolador que se va a utilizar, cual va a ser su frecuencia de reloj, cómo va a ser el circuito en que se va a utilizar el mismo, etc.

A diferencia de un programa de ordenador, donde uno escribe el programa, lo compila, lo ejecuta y ya, en el mundo de los microcontroladores hay que, previamente, definir el tipo de microcontrolador que se va a utilizar

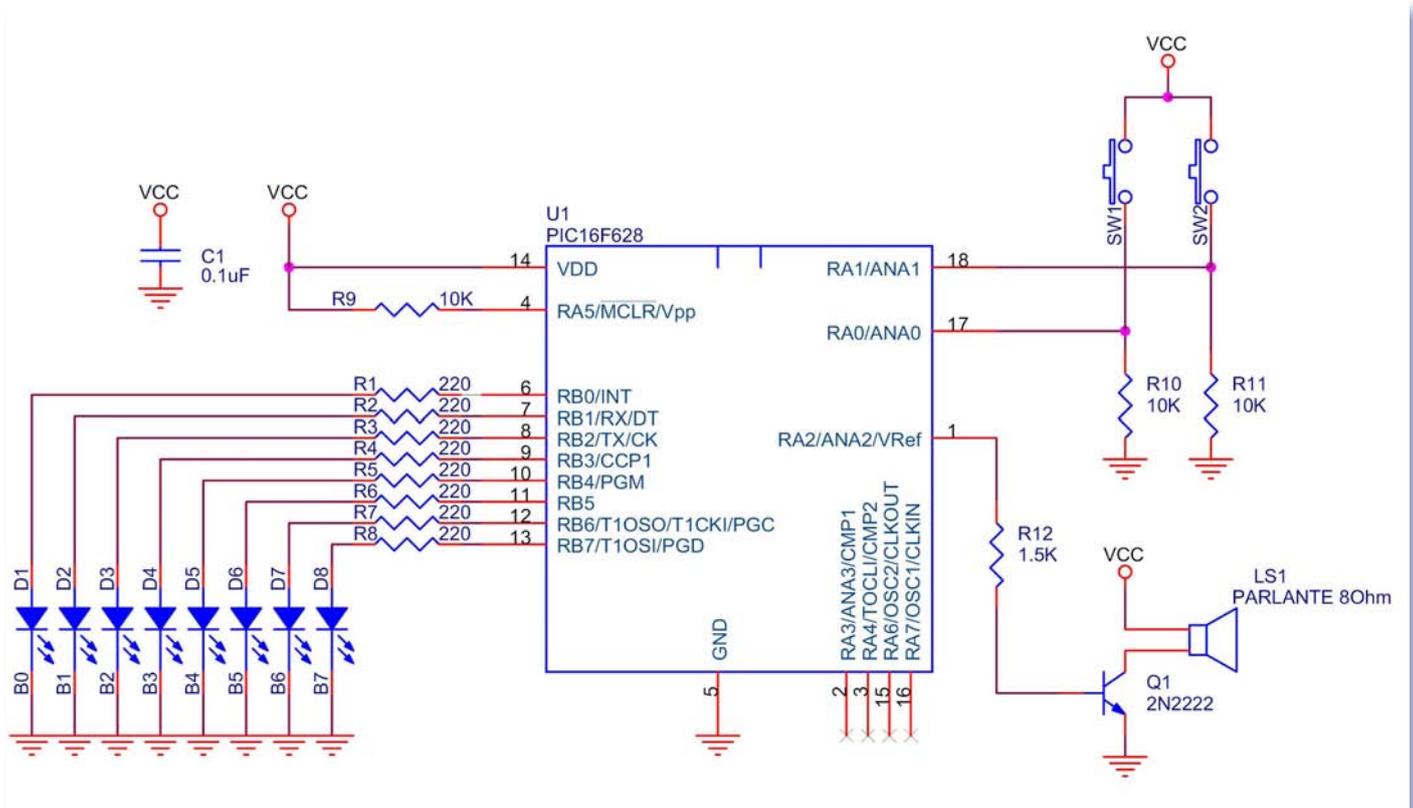
Para estas prácticas, utilizaremos un **PIC16F628A**, uno de los más difundidos y que más o menos viene a reemplazar al viejo y popular **PIC16F84A**, ya obsoleto.

El diagrama circuital que emplearemos para las primeras prácticas es el siguiente:

Si bien se supone que quien está leyendo este tutorial tiene una buena idea sobre electrónica y microcontroladores (si no, puede leer el resto de uControl), igualmente vamos a hacer una muy breve descripción del circuito.

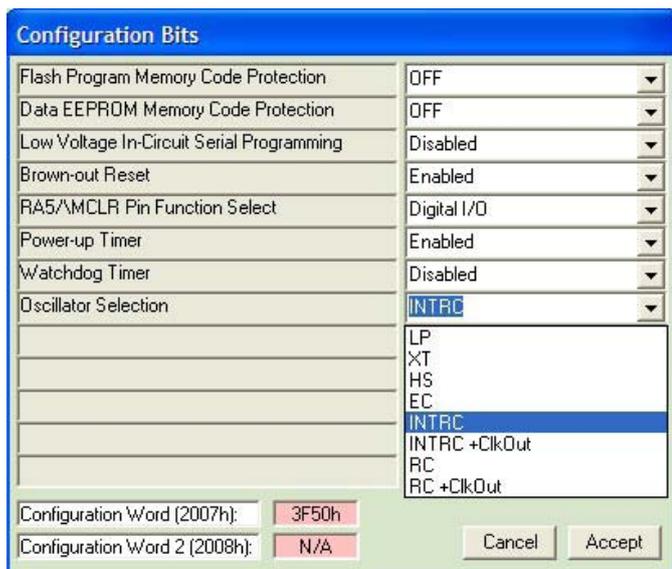
En primer lugar, vamos a aprovechar el oscilador interno del PIC16F628A y nos evitaremos el cristal y sus condensadores asociados. El puerto B del micro (pines 6 al 13) está conectado a 8 LEDs mediante 8 resistencias de 220ohms, que tienen como función limitar la corriente que circula por los LEDS. Estas serán nuestras "salidas". Los pines 17 y 18, correspondientes al PORTA.0 y PORTA.1 están conectados a pulsadores, que al ser presionados conducen 5V (un "1") al pin respectivo. Cuando están en reposo, las resistencias R1 y R2 se encargan de mantener el pin en "0". Por último, el pin 1 (PORTA.2) comanda un parlante mediante un transistor, para hacer alguna prueba con sonidos.

El circuito debe alimentarse con 5V bien filtrados y regulados.

Volviendo a nuestro programa, vamos a escribir el "hola mundo" de los microcontroladores: encender un LED.

El esquema de nuestro primer circuito.

El primer paso es, desde el menú "Opciones" -> "Select Microcontroller", elegir el PIC16F628A.

Lo primero es seleccionar el microcontrolador a utilizar.

Luego, debemos configurar los bits correspondientes:

Desde aquí configuramos los "fuses" del micro.

Lo destacable por ahora de esta configuración es que estamos dejando la memoria (FLASH y EEPROM) sin protección, que el pin RESET se va a comportar como I/O y que usaremos como oscilador el oscilador interno INTRC.

Una vez hecho esto, arrancamos el editor de BASIC (presionando CTRL-C, por ejemplo), y escribimos el siguiente código:

```
AllDigital
TRISA = %11111111
TRISB = %00000000
loop:
      PORTB.0 = 1
      WaitMs 500
      PORTB.0 = 0
      WaitMs 500
Goto loop
```

Vamos a analizarlo línea por línea para entender su funcionamiento:

La línea 001 utiliza la sentencia AllDigital para convertir todos los pines del micro en pines de E/S. Esto equivale a deshabilitar los comparadores, conversores A/D y todos los módulos que pudiese tener nuestro microcontrolador. No es la única manera de hacer esto, pero si la más sencilla desde el punto de vista del programador BASIC.

Las líneas 003 y 004 convierten todos los pines del puerto A en entradas (TRISA = %11111111) y los del puerto B en salidas (TRISB = %00000000). El "%" indica que el numero que viene a continuación esta en binario. Se podría haber escrito, por ejemplo TRISB = 0 y hubiera sido lo mismo. Personalmente prefiero el primer modo, ya que "veo" el estado de cada pin. Por supuesto, es valido activar como entrada algunos pines, y como salidas otros, haciendo algo parecido a TRISB = %11000111.

En la línea 006 encontramos una "etiqueta" (loop:). Esta no hace nada, solo sirve como referencia para enviar el flujo del programa a esa línea desde otro lugar, mediante la sentencia "Goto".

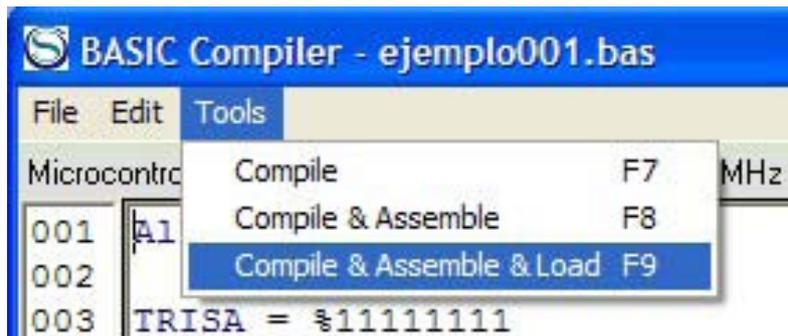
La línea 007 pone en "1" el pin correspondiente a PORTB.0, de manera que en el pin 6 del microcontrolador habrá 5V. Esta tensión hará que circule una corriente a través de la resistencia limitadora y el LED1, haciendo que este se encienda, ya que el cátodo se encuentra conectado a 0V.

En 008 tenemos la sentencia WaitMs 500. WaitMs se encarga de hacer una pausa en milisegundos. La duración de la pausa está dada por el número que sigue a la instrucción, en este caso 500 milisegundos, o medio segundo.

Luego, en 009, otra vez se vuelve a poner en 0 el pin 6, mediante PORTB.0 = 0 , lo que provoca que ese pin se ponga a 0V, y no haya más circulación de corriente a través de la resistencia y del LED, con lo que este se apaga.

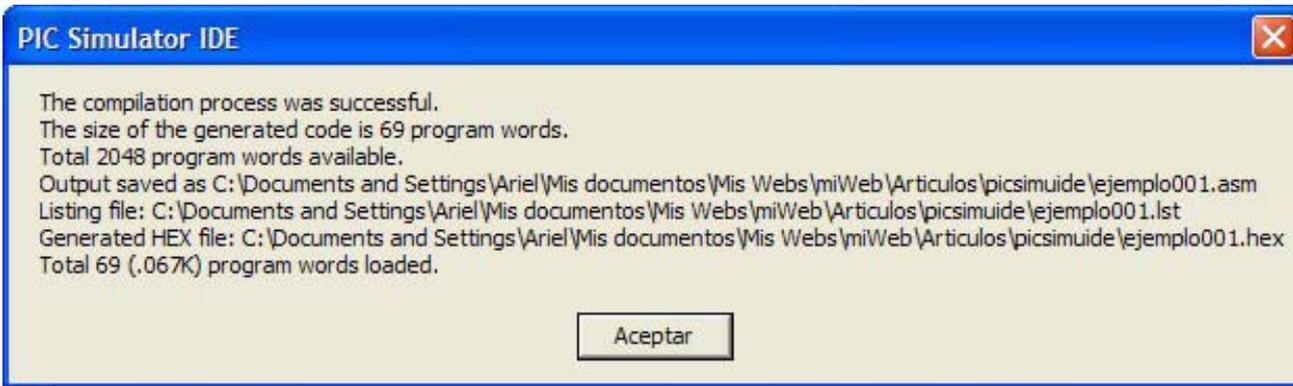
En 010 se hace nuevamente una pausa de medio segundo, y por último, la línea Goto Loop hace que el programa continúe en la línea 006 (que es donde está la etiqueta Loop).

El programa se repite indefinidamente, encendiendo el LED medio segundo, apagándolo otro medio segundo.

Si presionamos F9 o vamos al menú *Compile & Assemble & Load.*

Presionando F9 llegamos a este menú.

PIC Simulator IDE

The compilation process was successful.
The size of the generated code is 69 program words.
Total 2048 program words available.
Output saved as C:\Documents and Settings\Ariel\Mis documentos\Mis Webs\miWeb\Articulos\picsimuide\ejemplo001.asm
Listing file: C:\Documents and Settings\Ariel\Mis documentos\Mis Webs\miWeb\Articulos\picsimuide\ejemplo001.lst
Generated HEX file: C:\Documents and Settings\Ariel\Mis documentos\Mis Webs\miWeb\Articulos\picsimuide\ejemplo001.hex
Total 69 (.067K) program words loaded.

Aceptar

Este es el resultado de la compilación.

PIC SIMULATOR IDE compilara el programa, y cargara el archivo ".HEX" resultante en el simulador. Aparecerá el cuadro de dialogo donde se nos informa, entre otras cosas, si han ocurrido errores o no, el tamaño del programa (69 words), y la ruta a donde se ubicaron los archivos generados.

Si volvemos a la ventana principal del PIC SIMULATOR IDE, y desde "Tools" -> "Microcontroller View", abrimos la vista del microcontrolador. Al darle "Start" a la simulación tendremos algo parecido a lo que sigue:

PSI está simulando nuestro programa.

BASIC Compiler - ejemplo001.bas

```
001 AllDigital
002
003 TRISA = %11111111
004 TRISB = %00000000
005
006 loop:
007     PORTB.0 = 1
008     WaitMs 500
009     PORTB.0 = 0
010     WaitMs 500
011 Goto loop
012
013
014
```

En la captura se puede apreciar que el pin 6, correspondiente a RB0 está en "ON". Si esperamos lo suficiente, veremos cómo pasa a "OFF", y más tarde vuelve a "ON", etc. Si queremos esperar menos tiempo, y esto lo debemos tomar como una regla general al correr simulaciones, podemos disminuir el tiempo indicado en las instrucciones "WaitMS" a valores iguales a 1, de esta manera la simulación será mucho más ágil. Por supuesto, al momento de llevar el archivo ".HEX" a nuestro microcontrolador en el circuito real, debemos cambiar a los tiempos originales y volver a compilar. Caso contrario, el LED permanecería encendido solo una milésima de segundo, luego apagado el mismo tiempo, etc., por lo que nuestro ojo lo percibiría como encendido a medias, incapaz de discriminar su verdadero estado.

Se podría haber utilizado la instrucción SYMBOL para hacer más claro el programa. En el siguiente ejemplo, hemos hecho algunos cambios y obtenido un programa que hace exactamente lo mismo que el anterior, pero que resulta más claro de entender, ya que se aproxima algo más al lenguaje natural:

El mismo programa, pero utilizando "SYMBOL".

BASIC Compiler - ejemplo001.bas

Microcontroller: PIC16F628A; Clock Frequency: 4.0 MHz

```
001 AllDigital
002
003 TRISA = %11111111
004 TRISB = %00000000
005
006 Symbol led1 = PORTB.0
007 Symbol esperar = WaitMs 500
008
009 loop:
010     led1 = 1
011     esperar
012     led1 = 0
013     esperar
014 Goto loop
015
016
017
```

En BASCI con solo un puñado de instrucciones es posible crear un programa

.Conclusión:

En esta primera entrega solo hemos "arañado" la superficie de lo que el lenguaje BASIC puede ayudarnos en el desarrollo de nuestros proyectos. Como hemos visto, con solo un puñado de instrucciones es posible crear un programa, y lo mejor de todo, es que resulta más fácil de aprender que el lenguaje ensamblador.

En el siguiente número de uControl seguiremos aprendiendo a programar en BASIC. ¡Hasta la próxima! ■

control de volumen digital

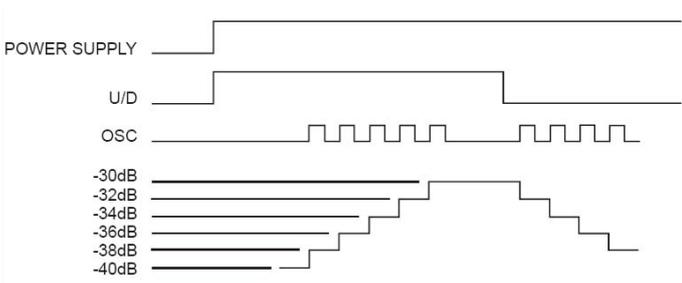
Démosle un toque profesional a nuestros proyectos con audio, aplicando controles de volumen accionados por pulsadores.
Digamos adiós a los ruidos molestos de nuestras consolas de mezcla con este verdadero potenciómetro digital.

Todos los potenciómetros mecánicos conocidos tienen, por lógica elemental, un desgaste que depende del uso o abuso que se haga de él. Sean rotativos o deslizantes, todos sufren este proceso, que trae aparejados ruidos molestos, cortes de sonido, subidas o bajadas abruptas de uno o ambos canales y tantos desagradables malos momentos, que podemos solucionar con este proyecto.

El corazón de nuestra realización es el CI PT2253A de Princeton Technology Corp., el cual nos ofrece mediante la utilización de dos pulsadores, bajar o subir el nivel de una señal de audio, tal como lo haríamos con el eje de un potenciómetro. Es un circuito integrado que posee muy baja distorsión de salida, puede ser alimentado con tensiones entre 6V y 12 V, en forma simple o simétrica y es capaz de controlar por pasos de 2dB desde -68dB hasta 0dB. Viene en un encapsulado DIL de 16 pines, incorporando en el mismo chip, el sistema completo para ambos canales de audio, además de ser el CI que actualmente se utiliza en la mayoría de los mini-componentes, para controlar el volumen.

La atenuación puede ser aumentada o decrementada, dependiendo del estado del pin 10, U/D (Up/Down), por la actuación del oscilador incorporado que posee el CI.

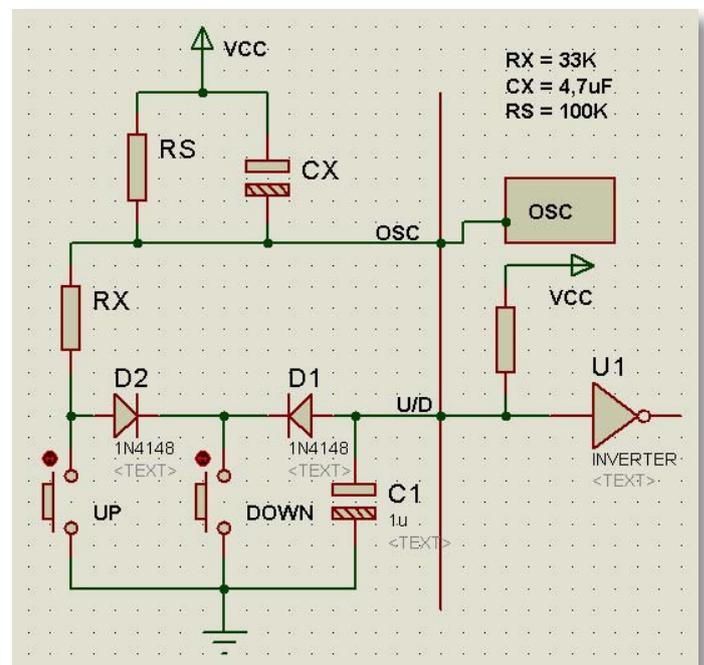
La frecuencia de este oscilador determinará la "velocidad" con que actuará nuestro potenciómetro digital.

El siguiente gráfico, extraído de la hoja de datos del IC, nos permite entender la importancia de una frecuencia de oscilación acorde a nuestras necesidades.

A mayor frecuencia de oscilación obtendremos una subida o bajada más rápida del volumen.

Para calcular la frecuencia de oscilación, debemos seguir el procedimiento según la siguiente fórmula:

```
Fosc = 1 / 0.7 * (CX) * (RX)
```

Esta fórmula nos dará un resultado expresado en Hz, considerando que Rs debe ser mayor o igual a 3 veces RX (3 * RX). Los valores de las resistencias se toman en Ohm y los valores de los capacitares en expresados en Faradios, para obtener el valor correcto.

La conexión básica del oscilador, en conjunto con los pulsadores, se realiza tal como lo muestra el circuito.

Circuito elemental de conexión oscilador/pulsadores.

Si colocamos la punta del osciloscopio en el pin 9, correspondiente al oscilador, notaremos que no tenemos oscilación alguna hasta que no pulsemos una de las teclas. Esto hace que el circuito en modo de espera, consuma una corriente despreciable.

En el caso de pulsar para subir el volumen (UP), el circuito del oscilador se cerrará a través de RX y el pro-

pio pulsador. De esta forma queda activado el oscilador, sin influir en absoluto en el pin U/D. Gracias a los diodos D1 y D2 y a la polarización interna de la resistencia de pull-up integrada en el chip, se mantendrá U/D en un estado lógico alto. Esto provocará que el IC interprete la instrucción de decrementar la atenuación, permitiéndonos subir el volumen.

Si por el contrario, pulsamos para bajar el volumen (DOWN), el circuito del pin oscilador se cierra a través de RX, D2 y el propio pulsador. Mientras que simultáneamente, el pin U/D pasa a un estado lógico bajo, a través de D1 y el propio pulsador DOWN. Esto será interpretado como que se debe aumentar la atenuación, provocando una disminución del volumen.

Vemos además en el gráfico, los valores elegidos para RS, RX y CX.

Cada vez que se inicialice la alimentación en el CI, éste adoptará una atenuación de -40dB, por defecto.

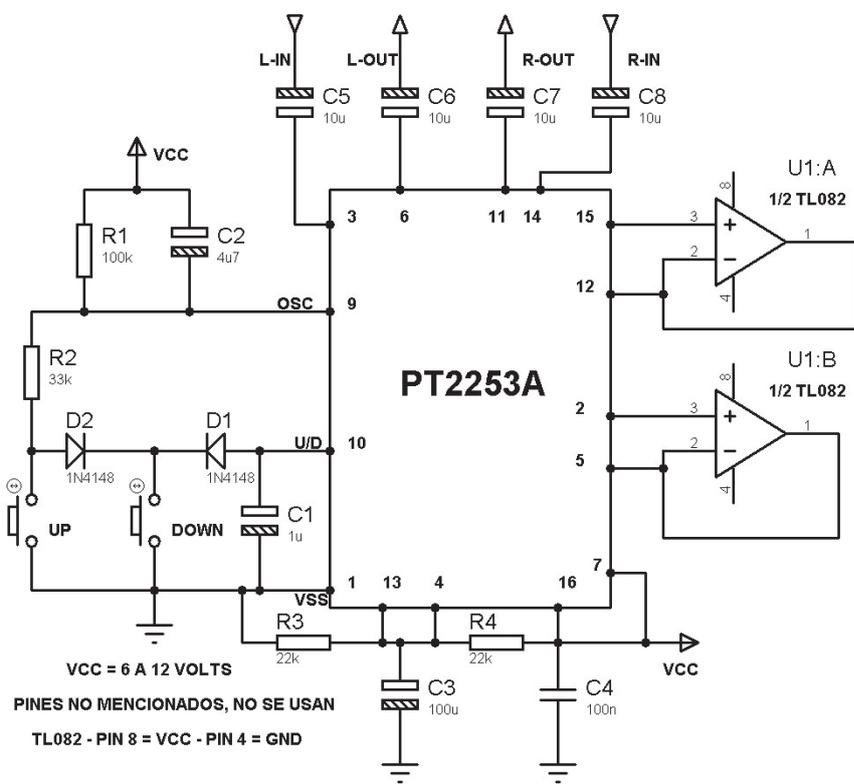
Cabe agregar, que por cada canal, el CI posee dos atenuadores, los cuales uniremos entre sí para lograr nuestro control y lo haremos a través de cualquier operacional doble, compatible con la serie de bajo ruido de los TL082, que en el circuito de la figura 1 aparece como U1. Esta unión de ambos atenuadores nos permite lograr los valores de -68dB a 0dB.

Ahora sólo nos resta elegir el tipo de alimentación que implementaremos.

Una alimentación simple de 9V ó 12V la encontraremos prácticamente en cualquier equipo al que queramos dotar con nuestro proyecto. Con esta elección, VSS estará a GND y los pines denominados Analog-GND, se alimentarán a través de un divisor resistivo como se ve en el circuito de la figura 2, formado por dos resistencias de 22K y un capacitor electrolítico de 100uF.

Como comentario final podemos agregar que no debemos olvidarnos de apantallar muy bien las conexiones de entrada y salida, así como también las partes internas de la placa donde realicemos nuestro circuito, para evitar ruidos captados por malas conexiones.

Una fuente bien filtrada … y listo! , nuestro potenciómetro digital, sustituyendo el ruidoso y viejo potenciómetro mecánico. ■

Diagrama completo del Potenciómetro Digital

registros de desplazamiento

A menudo debemos resolver situaciones en las que el número de salidas disponibles en el microcontrolador que estamos usando es insuficiente. Una manera sencilla de controlar varias salidas a partir de unos pocos pines consiste en la utilización de los llamados registros de desplazamiento. Pero ¿sabes qué son y cómo emplearlos?

Un registro de desplazamiento es una configuración circuital muy utilizada, generalmente para convertir un flujo de datos en forma serial a uno del tipo paralelo, motivo por el cual a menudo los chips encargados de esta tarea son llamados conversores serie-paralelo.

Por supuesto, es posible construir un registro de este tipo a partir de componentes discretos, aunque en la práctica resulta no solo inapropiado por cuestiones de tamaño y velocidad, si no también económicas, ya que un chip como los que mencionaremos en este texto rara vez supera el valor de $1.00 dólar.

La mejor manera de entender conceptos nuevos es apoyándose en analogías con temas que nos son familiares. En este caso no vamos a hacer una excepción, por lo que utilizaremos como ejemplo el funcionamiento de una cola, como la de un banco o la de una tienda cualquiera.

Supongamos que dos tipos de personas pueden formar parte de una cola. Estos dos tipos de personas son las que se ven en la figura siguiente, y es imposible confundir una con otra. Es decir, siempre estaremos seguros que en una posición determinada de la fila se ubica una u otra persona. Las llamaremos "0" (el "gordito") y "1" (al más "flaco"). Aclaro que la elección de los personajes solo tiene que ver con el parecido con el '0' y el '1'.

La cola que usaremos como ejemplo tiene 8 lugares, que hemos numerado del 0 al 7, pero nada impide trabaja con colas más largas, por lo que todo lo que se vea aquí, puede ser generalizado para colas de la longitud que se desee.

Figura 1: "0" y "1", nuestros personajes.

Otra particularidad de nuestra hipoté tica cola es que nunca puede estar vacía. Todas sus posiciones tienen que estar ocupadas, ya sea por "gorditos" o "flacos". En el estado inicial, la cola se encuentra completamente llena de "gorditos", como se observa en la Figura 3.

Figura 2: La cola utilizada como ejemplo tiene 8 posiciones.

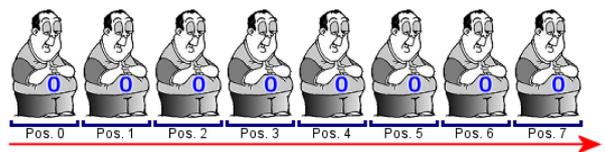



Figura 3: El estado inicial de la cola es este: completa de "gorditos".

Nuestra cola funciona como cualquier cola de la vida real: cuando alguien nuevo llega a la fila, se coloca en el lugar de más atrás, que en este caso corresponde a la "posición 0". Como nuestra cola tiene una longitud máxima de 8 posiciones, para hacer lugar al recién llegado, es necesario que todos los que estaban en la fila "avancen" una posición. El que estaba en la posición 0 pasa la 1, el que estaba en la 1 pasa a la 2, y así hasta llegar al que estaba en la posición 7, que "sale" por el extremo opuesto.

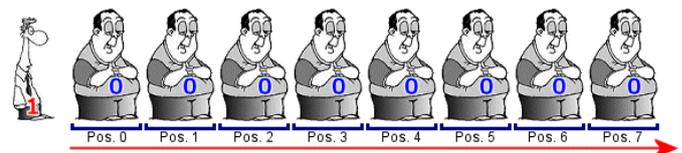



Figura 4: Llega un nuevo integrante a la cola....

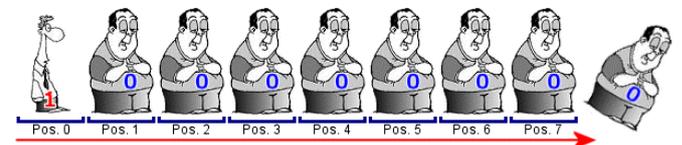



Figura 5: y ocupa el último lugar, desplazando a todos los demás una posición. El primero "sale" de la fila.

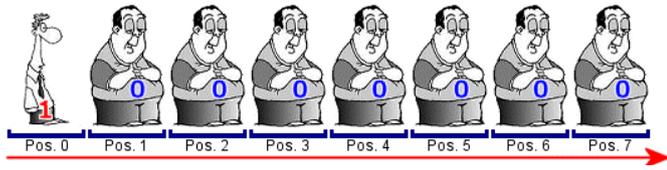



Figura 6: Este es el estado final de nuestra fila, con el nuevo integrante en el último lugar.

Si continuaran ingresando personas en la fila, el proceso se repetiría con cada nuevo integrante que llegue. Como el que entra primero es el primero en salir, a este tipo de colas se las llama **FIFO**, por First Input, First Output (Primero que entra, primero que sale).

Con todas estas cuestiones en mente podemos seguir avanzando en la comprensión del funcionamiento de los registros de desplazamiento. Supongamos que queremos que en la cola haya dos personajes flacos en los primeros lugares, luego un gordo, otra vez dos flacos, luego otro gordo por ultimo dos flacos más (como siempre, 8 personas en total). Sabiendo que cada personaje que ingresa en la cola desplaza a todos una posición a la derecha, si queremos que el que termine ocupando el extremo derecho de la cola sea un flaco, ese será el que primero debe entrar. Siguiendo el razonamiento anterior, los personajes deberían entrar en la fila en el orden siguiente:

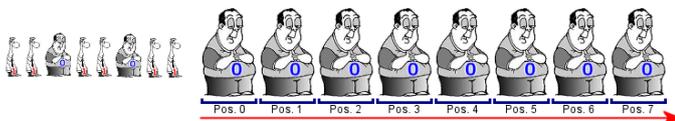



Figura 7: Los nuevos integrantes de la fila, esperando para ocupar su lugar.

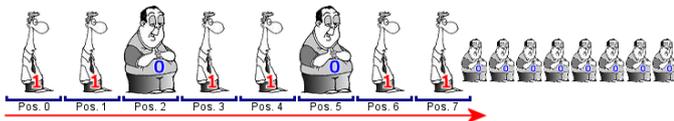



Figura 8: Este es el estado final de nuestra fila, con los integrantes originales desplazados hacia la derecha.

Poniendo fin a nuestra analogía, tendríamos que los integrantes de esta hipotética cola son los '0's y '1's (o estados altos y bajos) de nuestros circuitos, es decir, nuestros datos. La cola en si es el registro de desplazamiento. Cuando decíamos que el estado inicial de la cola eran 8 gordos, estábamos queriendo decir que al alimentar nuestro circuito, todas las salidas estarán en '0' o estado bajo.

Hay una salvedad, y es la existencia del reloj (clock en inglés). Efectivamente, en un circuito real, los datos pasan al registro de desplazamiento con cada pulso de reloj. Podemos pensar en este reloj como si se tratase de un "maestro de ceremonias", que da una palmada cada vez que alguien debe ingresar en la cola.

Muchos circuitos de registros de desplazamiento reales también incluyen un sistema de RESET, que permite poner simultáneamente todas las salidas en '0' sin necesidad de ingresar 8 ceros seguidos. Esto permite limpiar rápidamente el registro de desplazamiento.

.El 74HC164N

Existen varios circuitos integrados que implementan un registro de desplazamiento en su interior, por ejemplo, el muy conocido 74HC164N. Este interesante circuito integrado de la familia TTL viene en diferentes "sabores", de acuerdo a parámetros como: velocidad, temperatura de operación, voltajes y corrientes soportadas, entre otros. Dichas características dependen de las letras entre el "74" y el "164".

Desde el punto de vista técnico, dentro de este integrado se encuentra un registro de desplazamiento completo, de 8 bits de largo. Esto significa que se comporta como un conversor serie-paralelo, en el que se introducen pulsos de reloj por el pin CP y los datos en serie por los pines DSA y DSB, que son las entradas a una puerta AND. Las 8 salidas van tomando los estados indicados por el "tren" de datos de la entrada. Dispone de un pin (el 9) que realiza la función de poner en cero todas las salidas (RESET, MR en el esquema de la Figura 9).

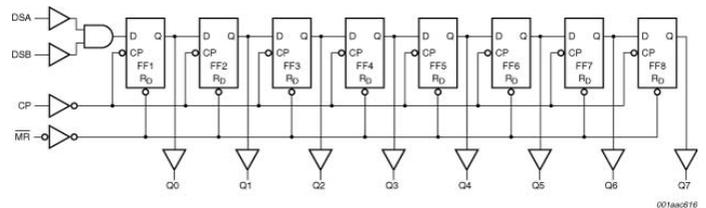



Figura 9: Esquema interno del integrado 74LS2164N

Como puede verse en la imagen correspondiente a su pinout, el integrado dispone de solo 14 patillas. Los pines 1 y 2 son la entrada de datos. Como dijimos antes, internamente existe una puerta AND que realiza el producto lógico de los valores de ambas entradas. En general, se unen entre sí para que el resultado de la función AND sea igual al valor del dato, o bien se pone una de las entradas en alto (conectándola a +5V) para que la otra entrada sea la que determine el valor de la salida. Cualquiera de las dos formas es válida. Por supuesto, existen aplicaciones donde se obtienen datos de dos fuentes distintas, en cuyo caso se conectará una entrada de la puerta a cada una de las entradas de datos.

Los pulsos de clock entran por el pin 8. Los datos de la entrada se reflejan en la salida con cada transición bajo-alto del reloj. Los pines 3,4,5,6 ,10,11,12 y 13 son, en ese orden, las salidas.

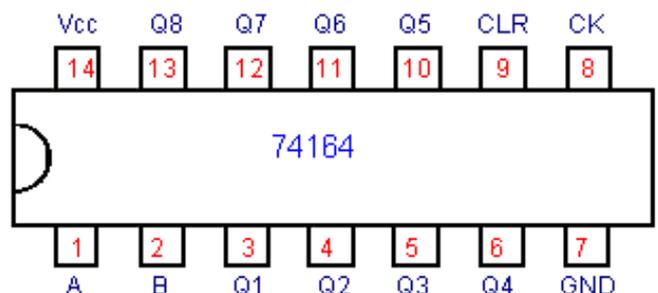



Figura 10: Función de cada patilla del 74LS2164N

CD4094 primera parte

para manejar displays 7 segmentos

Siguiendo con el análisis comenzado en el artículo **Registros de desplazamiento** publicado en este mismo número, se estudiará el circuito integrado CD4094, de Fairchild semiconductor, que posee prestaciones similares al 74HC164N.

El **CD4094** es un registro de desplazamiento de 8 bits, con salida tri-estado. Esto significa que disponemos de un mecanismo para aislar sus 8 pines de salida del resto del circuito. Los datos son desplazados serialmente con cada flanco de subida del reloj **(CLOCK)** y cada bit es transmitido al latch correspondiente con cada flanco de bajada del pin **STROBE**

Las características más destacables de este circuito integrado son:

- *Rango de voltaje: 3.0 V a 18.0 V*
- *Compatibilidad con la familia TTL*
- *Salida tri-estado*

Diagrama de Conexión del CD4094.

¿Por qué usar un registro de desplazamiento para un display 7-segmentos? Es muy sencillo: nos ahorramos varias líneas de entrada/salida. En el caso de usar un solo display se ahorran 5 pines, ya que se necesitan solo 3 líneas para controlar los 7 segmentos y el punto decimal. Y si necesitamos controlar un número mayor de display, el ahorro es mayor: las mismas tres líneas usadas para el control de un display permiten controlar todos los que necesitemos. ¿Interesante, verdad?

Diagrama de tiempos del CD4094

Para que el **CD4094** pueda manejar un display 7 segmentos, tienen que haber unas señales de control establecidas, que podemos gestionarla con un microcontrolador. En este artículo usaremos el PIC 16F84A.

Para establecer las señales de control en el CD4094, es necesario estudiar su diagrama de tiempos:

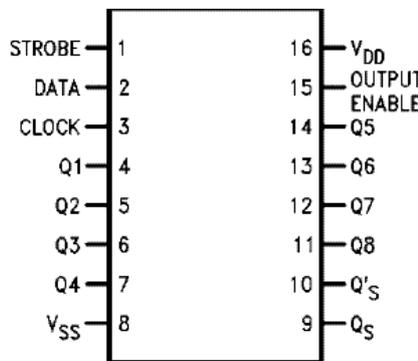
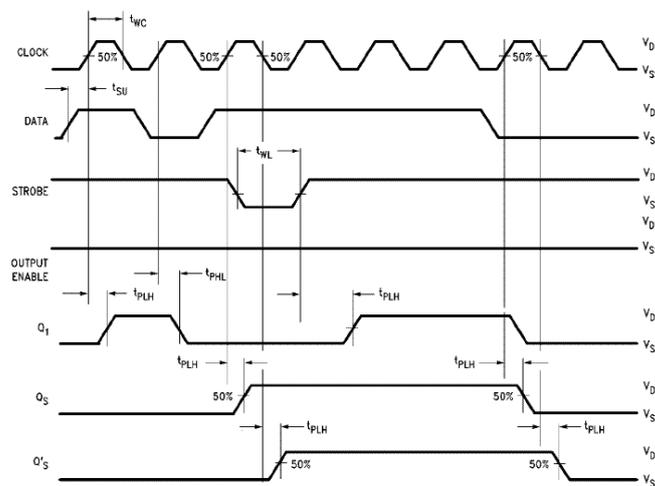
La primera señal de control es **DATA,** que es el dato que enviaremos serialmente. Enviamos un flanco ascendente y luego un flanco descendente. En el instante que el **CLOCK** esté subiendo, el estado del **DATA** será reconocido como un dato válido por el CD4094

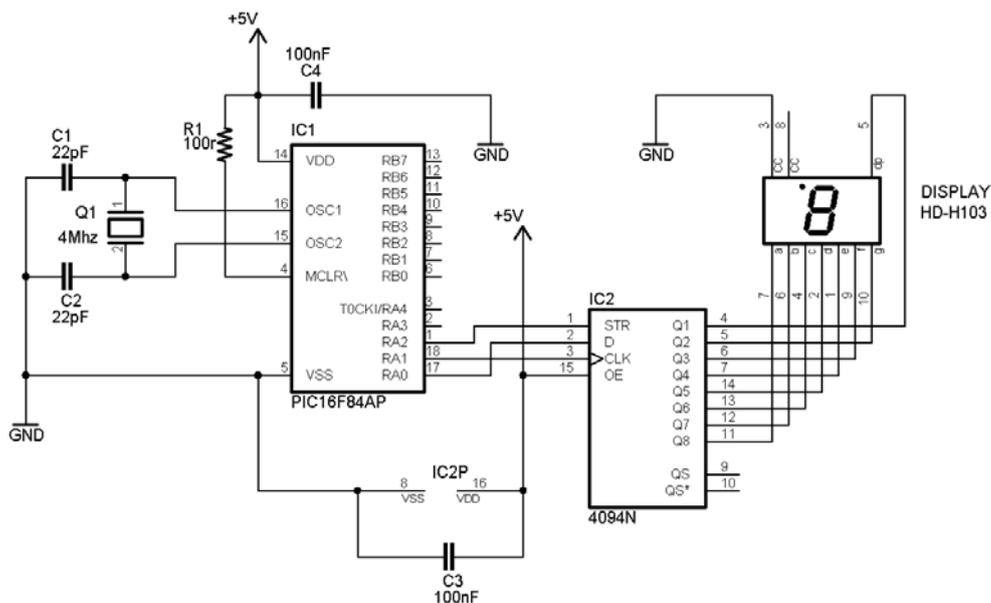
La señal de control **STROBE** se usa en caso de que queramos tener en los latch de salida los bits enviados por el PIC a medida que se van recibiendo.

Como se usará un display 7 segmentos, lo mejor es ver el dato completo una vez finalizada la transferencia.

Entonces **STROBE** en un nivel alto mientras enviamos los datos. Con ello garantizamos que el display no muestre valores "extraños" hasta que no pongamos **STROBE** en bajo, instante que se transferirán los 8 bits completos a los latches de salida.

Por último tenemos el pin **OUTPUT ENABLE.** Este pin cumple la función del tercer estado del latch, el estado de alta impedancia. En este caso no lo usaremos. Generalmente se emplea en sistemas de buses, donde se conectan varios integrados a las mismas señales, permitiendo aislar un circuito de otro. Así que mantendremos este pin en un nivel alto.

Una vez comprendido el funcionamiento del esquema de tiempos del CD4094, se procederá a escribir un programa para el microcontrolador PIC16F84A usando el compilador de lenguaje C de la empresa CCS:

```
// Usando el CD4094 para manejar 1 display 7 segmentos


#include <16F84A.h>
#fuses XT,NOPROTECT,NOWDT,PUT
#use delay(clock=4000000)

// Definición de pines de control
#define DATA PIN_A0
#define CLOCK PIN_A1
#define STROBE PIN_A2

// tabla de numeros constantes que contienen el correspondiente valor
// en 7 segmentos
int   const segmentos[16]={0x3f,0x6,0x5b,0x4f,0x66,0x6d,0x7d,0x7,
                           0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};

// declaración de la función escribir_4094
void escribir_4094(int caracter);

// programa principal
void  main(){
   output_a(0x0);
   output_b(0x0);
   set_tris_a(0x0);
   set_tris_b(0x0);

  while(true){
      for(t=0;t<16;t++){
         escribir_4094(t);          // envía el números en hexadecimal al display 7-seg
         delay_ms(1000);
      }
   }
}

//****************************************************************
// Función escribir_4094
//
// Descripción: envía serialmente el datode 8 bits de acuerdo al
// argumento caracter y activa las señales de control establecidas
// en el mapa de tiempos del CD4094
//
// datos de entrada: caracter un número entero
// datos de salida: ninguno
//****************************************************************
void escribir_4094(int caracter){
int contador_8;
   output_low(PIN_A0);
   output_low(STROBE);
   output_low(CLOCK);
   for(contador_8=0;contador_8<8;contador_8++){
      output_bit(DATA,bit_test(segmentos[caracter],contador_8));
      output_high(CLOCK);
      delay_cycles(1);
      output_low(CLOCK);
   }
   output_low(CLOCK);
   output_high(STROBE);
   delay_cycles(1);
   output_low(STROBE);
}
```

.El esquema eléctrico:

esquema del circuito eléctrico

Nota: hay que tomar en cuenta la velocidad de operación del CD4094 y que las señales de control no deben ser menores al especificado por la hoja técnica (datasheet). En nuestro ejemplo estamos trabajando con una frecuencia de oscilación de 4Mhz, esto es, 1uS por cada paso del contador de programa del microcontrolador, suficientes para los 0,666 uS mínimos que exige el CD4094 a 5V de alimentación.

Esto hay que considerarlo si se va a trabajar a niveles de tensión ó frecuencias muy altas■

Esta es la lista de componentes necesarios para el montaje:

Cantidad	Referencia	Modelo
1	IC1	PIC16F84A
1	IC2	CD4094
1	Q1	Cristal de 4Mhz
1	DISPLAY	HD-H103
2	C1,C2	22pF
2	C3,C4	100nF
1	R1	100r

.Bibliografía consultada:

Fairchild Semiconductor (1987). **CD4094BC 8-Bit Shift Register/Latch with 3-STATE Outputs. Datasheet.** *Disponible en: www.fairchildsemi.com*

.Glosario:

Latch: especie de buffer o memoria intermedia que almacena un estado binario y no cambia a menos que se le indique.

Tri-Estado : tres estados lógicos permitidos por una compuerta digital estado alto, estado bajo y alta impedancia
dash point: punto decimal que traen los displays 7 segmentos.

sensor de humo con LED y LDR

Una de las maneras posibles de detectar la presencia de humo se basa en comprobar la transparencia (o falta de ella) del aire. El proyecto que presentamos aquí tiene como función comprobar continuamente la luz que recibe una LDR de un LED ubicado enfrente, y en caso de que este valor disminuya, se cierran los contactos de un relé durante un cierto tiempo.

Este proyecto puede resultar interesante para estudiantes y hobbystas, pero considero que para su uso en un sistema de seguridad es insuficiente, ya que no se tiene en cuenta la posibilidad de que, por ejemplo, se hayan "pegado" los contactos del relé o la LDR haya dejado de funcionar. Es por esto que uControl no se responsabiliza por posibles fallas o pérdidas debidas al empleo de este circuito.

.El circuito

El circuito es sumamente simple: un LED, cuya corriente hemos limitado con una resistencia de 1K en serie, ilumina permanentemente a una LDR que se ubica enfrente. Esta LDR junto con el resistor ajustable de 50K configura un divisor de tensión que dispara el circuito integrado NE555 cuando la LDR está a "oscuras". El nivel del disparo se ajusta mediante ese resistor ("SENSIB" en el esquema) de manera de hacer más o menos sensible el sistema.

El circuito integrado NE555, configurado como monoestable, permanece en reposo mientras la LDR está iluminada. Cuando esta situación cambia, su valor disminuye y el circuito se dispara. Su salida permanece en estado alto un tiempo, que está determinado por valor el condensador electrolítico de 100uF/16V y la resistencia ajustable de 100K ("TIEMPO" en el esquema).

La salida del NE555, a través de una resistencia de 1K5 excita el transistor 2N3904, que a su vez permite que la bobina del relé se energice. Cuando el NE555 vuelve al estado de reposo, el transistor vuelve a bloquearse y el relé se desactiva. El diodo en paralelo con la bobina del relé está para evitar que la corriente generada en la desconexión dañe el transistor.

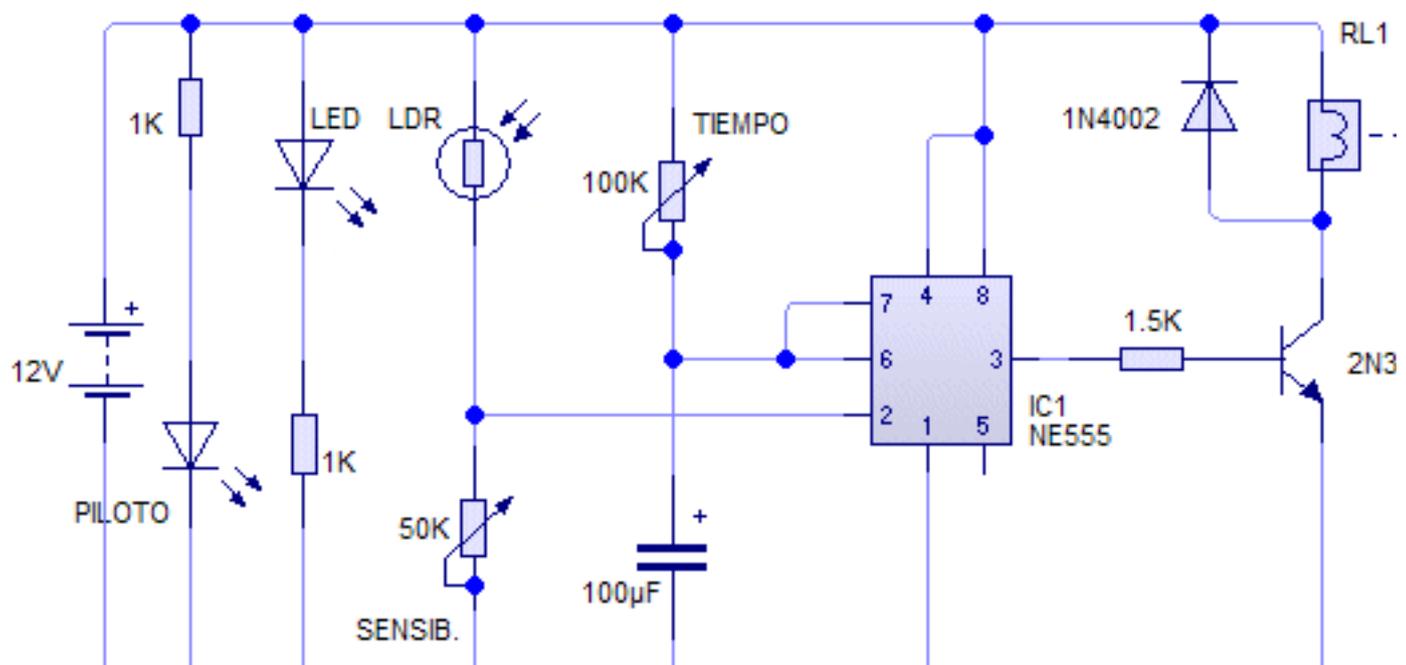



Figura 1: Este es el circuito eléctrico del detector de humo.

El relé es del tipo inversor, así que podemos elegir conectar el tipo de alerta deseado en el borne “normal abierto” (una sirena, por ejemplo) o “normal cerrado” (una luz que se apagaría en caso de incendio).

Hemos agregado un segundo LED, (“PILOTO” en el esquema) para que sierva como indicador de que el circuito está alimentado, por si la disposición del otro LED y la LDR son tales que no pueden verse con facilidad.

En el esquema se ve una batería de 12V alimentando el circuito, lo que lo hace apto para su uso aun en cortes de energía. Los que no quieran gastar en baterías, pueden conectar este circuito a una fuente de alimentación que entregue entre 9V y 12V de corriente continua bien estabilizados.

Montaje y PCB

Para que el armado del circuito sea lo más fácil posible, hemos dibujado un circuito impreso y también un esquema con la posición de los componentes sobre la placa. Puedes ver el PCB en la figura 2 y el la posición de los componentes en la figura 3.

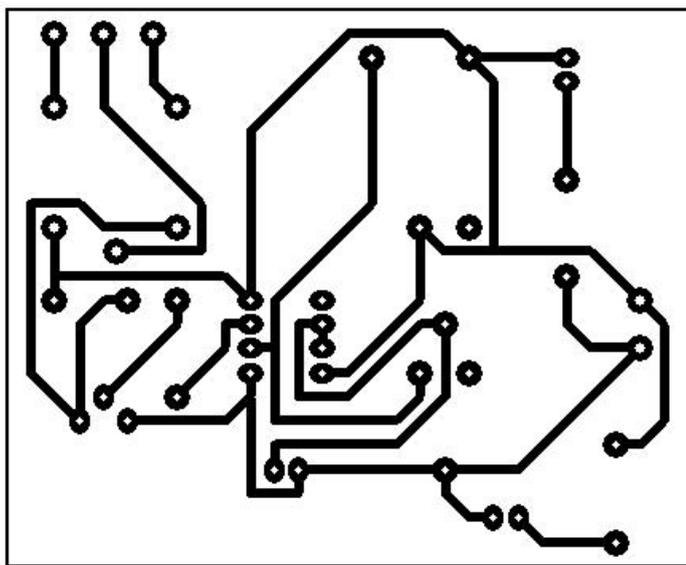



Figura 2: Este es el PCB a utilizar. *Descargalo de www.ucontrol.com.ar*

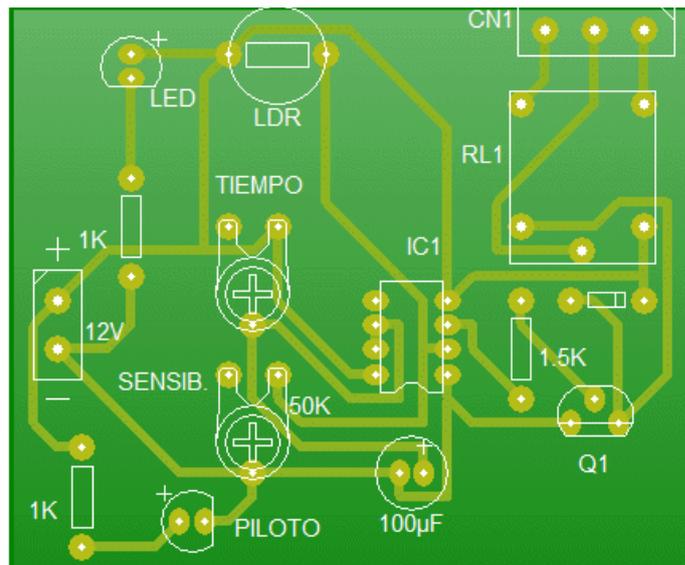



Figura 3: Puedes usar esta imagen para facilitar el montaje.

temporizadores programables

El tiempo, la cuarta dimensión, está presente en todo momento, y medir su paso es un aspecto importante en casi cualquier aplicación electrónica. Una de las primeras aplicaciones desarrolladas en el mundo de la electrónica digital fueron los circuitos para medir el tiempo, y estos adquirieron tal importancia, que están presentes en casi cualquier dispositivo electrónico moderno. En nuestro caso, hablaremos de un grupo especial de estos dispositivos, los circuitos temporizadores de los microcontroladores.

El primer microcontrolador de la historia, el INTEL 8051, que salió al mercado en 1976, tenía dos temporizadores programables. Hasta el día de hoy, estos han tenido una evolución constante en el mundo de la electrónica digital, de modo que algunos microcontroladores modernos poseen varios temporizadores.

.¿Cómo trabaja un temporizador?

El elemento fundamental del temporizador es un contador binario, encargado de contar los pulsos suministrados por algún circuito oscilador, con una base de tiempo estable y conocida.

El simple hecho de contar pulsos de una duración fija nos permite medir el tiempo con precisiones asombrosas, determinadas fundamentalmente por la estabilidad del generador de pulsos y por los circuitos electrónicos del contador binario. Sin embargo, un contador útil debe tener más elementos que permitan sacar provecho a ése circuito básico, es por ello que los microcontroladores utilizan un conjunto de circuitos auxiliares para poder manejar, con cierto nivel de libertad, las características básicas del contador binario y convertir el conjunto en un temporizador/contador programable.

.Se utilizan en…

Hacer una lista completa es prácticamente imposible, pero algunos ejemplos de su aplicación, nos ayudarán a adentrarnos en los entresijos de su diseño y sacarle provecho a sus potencialidades.

- Medición de tiempo
- División de frecuencia
- Medición de período y frecuencia
- Conteo de eventos
- Base de tiempo para otros periféricos
- USART
- PWM
- Watch Dog
- …

.Estructura básica

La estructura básica de un temporizador/contador la podemos ver en la Figura 1. En este esquema simplificado podemos observar que el contador está compuesto por tres bloques fundamentales:

1.Contador binario: es el elemento básico del temporizador/contador y su misión es contar los pulsos del reloj. Hay dos propiedades esenciales a tener en cuenta, respecto a este componente: la cantidad de pulsos que puede contar y la posibilidad de controlar el sentido del conteo, sea ascendente o descendente.

2.Circuitos de configuración y control: constituyen la interfaz entre el contador binario y los circuitos externos. Es uno de los elementos que da valor añadido al simple contador binario.

3.Circuitos especializados de salida: Se utilizan para notificar, a otro elemento del sistema, sobre el estado del temporizador o acerca de la ocurrencia de un determinado evento.

Más adelante veremos, con ejemplos de contadores reales, cada uno de los bloques del temporizador/contador, y cómo configurar estos módulos, para utilizarlos en distintas aplicaciones.

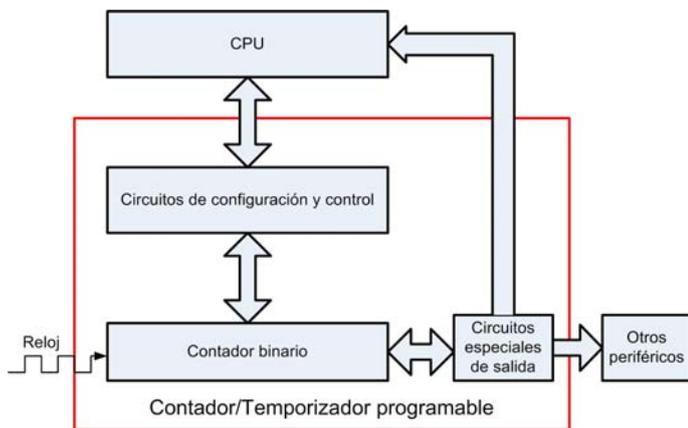



Figura 1: Estructura básica de un temporizador/contador

.Características

En el mundo del diseño digital, los temporizadores constituyen periféricos muy útiles. Se diseñan con ciertas características que determinan el uso que podemos darle a un temporizador, veamos algunas:

Longitud del contador: Los más comunes son aquellos que tienen 8 ó 16 bits, determina la cantidad máxima de pulsos que se pueden contar.

Lectura/escritura: En general, los temporizadores pueden ser escritos o leídos por el procesador del microcontrolador. En algunos casos, donde el temporizador está vinculado a algún periférico muy específico, esta opción puede no existir o estar limitada.

Modos de trabajo: Existen, en principio dos: contador y temporizador. Como contador, se cuentan los pulsos desde una fuente externa al microcontrolador. Los pulsos contados pueden tener período variable. Como temporizador, se cuentan los pulsos suministrados por una fuente estable y conocida, que puede ser externa, o alguna fuente generada internam controlador.

La forma típica es el conteo ascendente, sin embargo, existen contadores con la opción de configurar el modo de conteo, sea éste ascendente, descendente o de otro tipo específico.

Configuración de activación por frente: Permite establecer cuando se produce el conteo, si en el frente de subida del reloj o en el frente de caída del mismo.

Configuración del reloj: En la mayoría de los casos la fuente de reloj es configurable. Incluso existen microcontroladores con abundante variedad de formas de configurar el reloj, de modo que se puedan obtener distintos tipos de bases de tiempo.

Interrupciones: En la mayoría de casos, los temporizadores tienen interrupciones asociadas, con el objetivo de notificar al procesador que ha ocurrido el cruce por cero o algún valor específico en el registro de conteo.

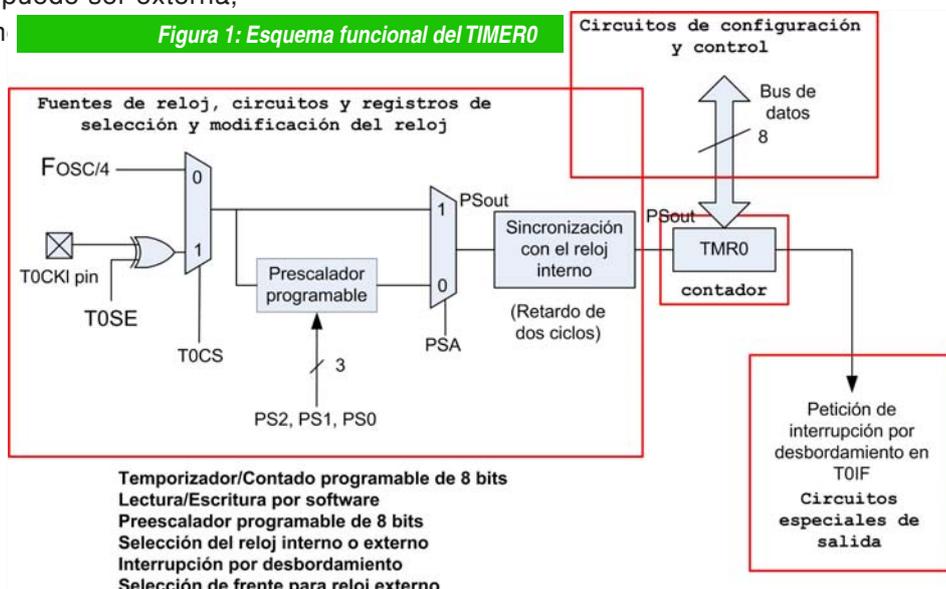
Características especiales: Muchos microcontroladores tienen temporizadores especializados para trabajar de conjunto con algunos de sus periféricos, o para ser utilizados en aplicaciones específicas.

.Un temporizador por dentro

Ahora vamos a analizar un temporizador real, en particular, el TIMER0 de los microcontroladores PIC de la MICROCHIP. Específicamente los PIC de gama media.

El análisis de este temporizador nos permitirá ver en concreto cómo hacer funcionar uno de estos dispositivos. Lo aprendido aquí puede ser muy útil en caso de trabajar con temporizadores de otros fabricantes, ya que si observa las hojas de datos de otros microcontroladores, podrá percatarse de la similitud de los diferentes módulos temporizadores entre distintos fabricantes.

El esquema funcional del temporizador podemos verlo en la Figura 2, aquí hemos marcado con cuadros rojos lo que sería, más o menos, cada bloque de los explicados para la figura 1. Con un temporizador real no podemos hacer una separación completa de los elementos de cada uno de los bloques que usamos para introducir la teoría de los temporizadores, porque los circuitos y registros están muy relacionados entre sí para constituir un temporizador real.

Figura 1: Esquema funcional del TIMER0

Analizando el esquema de este temporizador podemos observar que el mismo puede utilizar como fuentes básicas de reloj una fuente externa conectada al pin T0CKI (el pin específico en el encapsulado del microcontrolador depende del dispositivo) o puede utilizarse la señal de reloj interna, generada por el microcontrolador, equivalente a un ciclo de máquina (FOSC/4).

La selección de una u otra fuente básica depende del estado de un multiplexor digital cuya señal de control es T0CS, a la salida de este multiplexor encontramos un pre-escalador, que sirve también como post-escalador para el Watch Dog (WD).

Un pre-escalador o post-escalador, es un contador cuya base de conteo es configurable, que se coloca antes o después del contador principal del temporizador y su función es dividir la señal del reloj a la entrada o salida del contador principal. Generalmente no se permite leer o escribir su registro de conteo.

Existe otro multiplexor digital que nos permite seleccionar o no el uso del pre-escalador y cuya señal de control es PSA. A su salida existe un componente que tiene como misión sincronizar el reloj externo con el del dispositivo, en caso que se utilice esta fuente, y al final tenemos la fuente de reloj que será la que hará funcionar al contador principal del temporizador.

El registro del contador principal del temporizador puede leerse y escribirse por software en cualquier momento, sin embargo, existen ciertos detalles a tener en cuenta en caso de las escrituras. A la salida del contador principal tenemos la activación de la señal T0IF, que es utilizada para notificar al procesador que se ha desbordado el temporizador, y en caso de estar habilitadas las interrupciones del dispositivo, esta señal activará el proceso de interrupción del procesador.

Para poder configurar este periférico y obtener información útil sobre él, existen tres registros que el programador y el periférico utilizan para lograr un trabajo adecuado:

- **TMR0:** es el registro donde se lleva la cuenta de pulsos de reloj que llegan al contador principal del temporizador, es de lectura/escritura.

- **INTCON:** Es el registro de control de interrupciones, se utiliza para habilitar el servicio de interrupciones del microcontrolador y del temporizador, contiene además la bandera T0IF, que determina si el temporizador se ha desbordado.

- **OPTION_REG:** Se utiliza para establecer la fuente de reloj a utilizar, el frente que activa el conteo y para configurar el pre-escalador.

.Concluyendo

Hasta ahora hemos visto una breve introducción a la teoría de los temporizadores y analizado el esquema funcional y los registros de trabajo del TIMER0 de los microcontroladores PIC, es importante que el lector dedique tiempo a revisar las hojas de datos y tome nota de las características de este y otros temporizadores, específicamente para el modelo de dispositivo que pueda tener.

Así podrá conocer con mayor nivel de detalle a este periférico.

Nota del editor:

En la próxima entrega tendremos un ejemplo concreto en el que pondremos a trabajar el TIMER0 para implementar un reloj, y utilizarlo en una de las aplicaciones típicas: un reloj digital.

Le sugerimos al amigo lector que tenga a mano las herramientas de su elección para poner a punto su reloj con temporizador versión 0.

control de velocidad de motores CC por PWM con NE555

Este proyecto es un buen ejemplo de aplicación práctica para el circuito integrado NE555. Se basa en el uso de este integrado en su configuración como oscilador astable, en la cual, el circuito produce en su pin de salida OUTPUT (3) una onda cuadrada, con amplitud igual a la tensión de alimentación y tiempo del estado alto ajustable.

La modulación de ancho de pulso, del inglés Pulse Width Modulation (PWM), consiste en variar a voluntad, el tiempo del estado lógico alto de una señal digital de período fijo. Este tipo de señal se puede obtener mediante distintas configuraciones de circuitos, y tienen amplia utilización práctica, un ejemplo es el control de velocidad de motores.

En nuestro caso, utilizaremos un CI NE555, para obtener en su pin de salida una señal PWM, lo cual nos permitirá controlar, en lazo abierto, la velocidad de giro de un motor de corriente continua (CC). Para ello utilizaremos una variante de circuito, en la cual el CI NE555, se utiliza en su configuración astable, pero que nos permite variar el tiempo de estado alto de la señal de salida.

En la configuración astable, la salida del CI NE555, no permanece fija en ninguno de los dos estados lógicos; si no que durante un tiempo está en estado bajo y durante otro cambia al estado alto. Estos cambios, se producen, en un tiempo que llamaremos T.

El periodo de tiempo (T) de la señal de salida es igual al la suma del tiempo en estado alto (Tm), del inglés Mark Time, y tiempo en estado bajo (Ts), del inglés Space Time. En general, en lugar de utilizar T como parámetro, utilizaremos la frecuencia (F) de la señal de salida, cuya expresión es: $F = 1/T$.

Nota del editor: *En el próximo número, publicaremos un artículo donde se explica en detalle el uso del CI NE555.*

.El circuito

Actuando sobre el potenciómetro R2 (Figura 1) se modifica la tensión presente en el pin 2 del CI NE555, lo cual produce un cambio en el tiempo de disparo. Para este circuito, T es de tamaño fijo, lo único que se hace al rotar R2, es cambiar Tm y Ts.

El diodo D1 evita que la corriente generada por el motor cuando está girando sin alimentación, destruya el transistor Q1. Este circuito se aplica perfectamente a pequeños motores de corriente continua, de entre 6V y 12V, con un consumo de corriente no mayor de 300 mA.

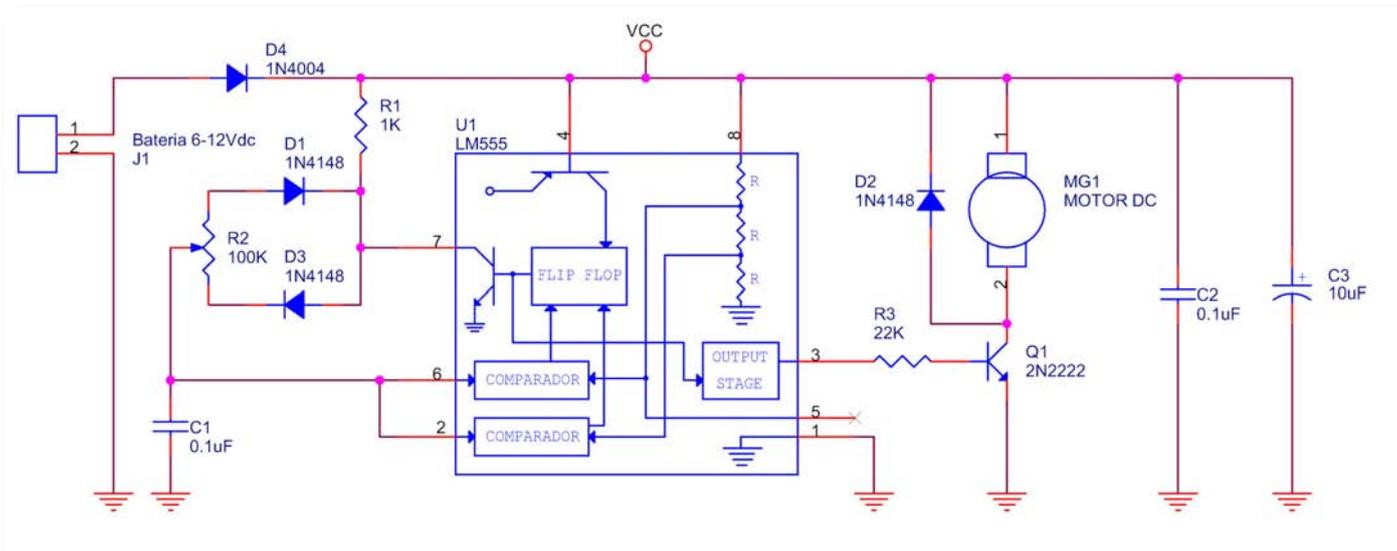



Figura 1: Circuito eléctrico de nuestro proyecto

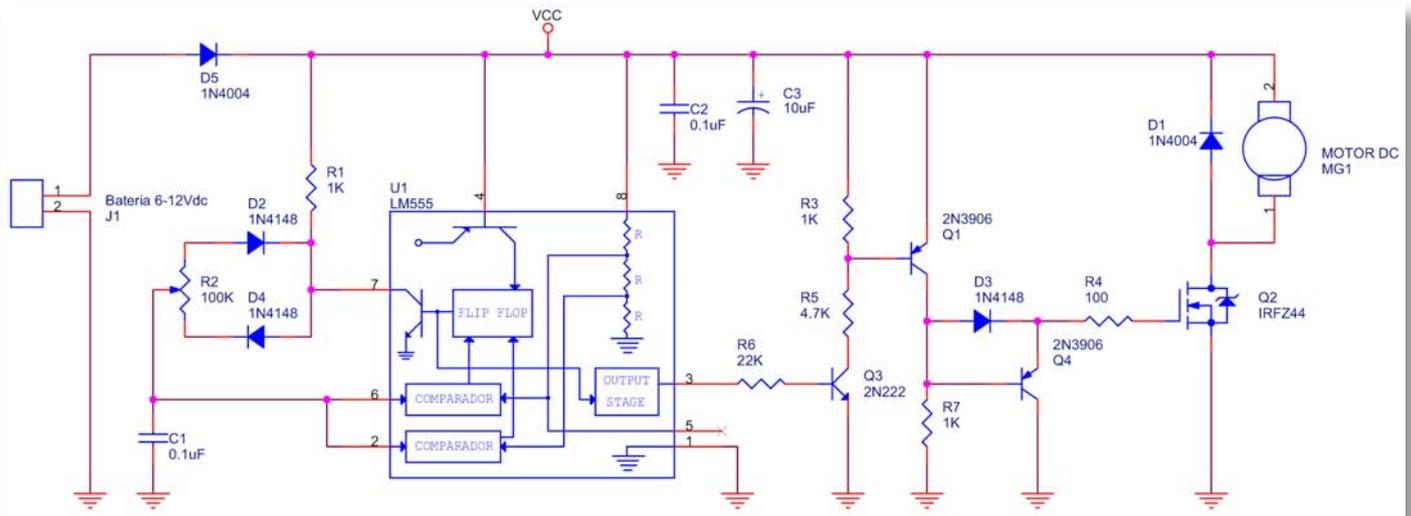



Figura 2: Circuito con MOSFET, para motores de mayor consumo de corriente

Para motores más grandes, se debe sustituir el transistor Q1 por uno de tecnología MOSFET, capaz de manejar picos de hasta 50A. Lo hemos probado con motores de 12V/3A y funciona sin calentarse, aún sin el uso de disipador de calor. El circuito para este tipo de motores puede observarse en la **Figura 2.**

.Funcionamiento del sistema de control con PWM

La **Figura 3** muestra el aspecto que tiene la señal de salida del circuito integrado (pin 3) cuando la resistencia variable R2 tiene el valor más alto. En este caso Tm es tan pequeño que el motor permanece detenido.

En el otro extremo, al llevar al potenciómetro a su valor mínimo, Tm adquiere su valor máximo.

La forma de onda para este caso, se muestra en la **Figura 4.**

Con un ancho de pulso del 100%, Tm ≈ T, el motor recibirá alimentación de CC casi todo el tiempo, y girará a su máxima velocidad. Cualquier punto intermedio entre estos dos estados es válido. Logrando, de este modo, controlar la velocidad de giro del motor.

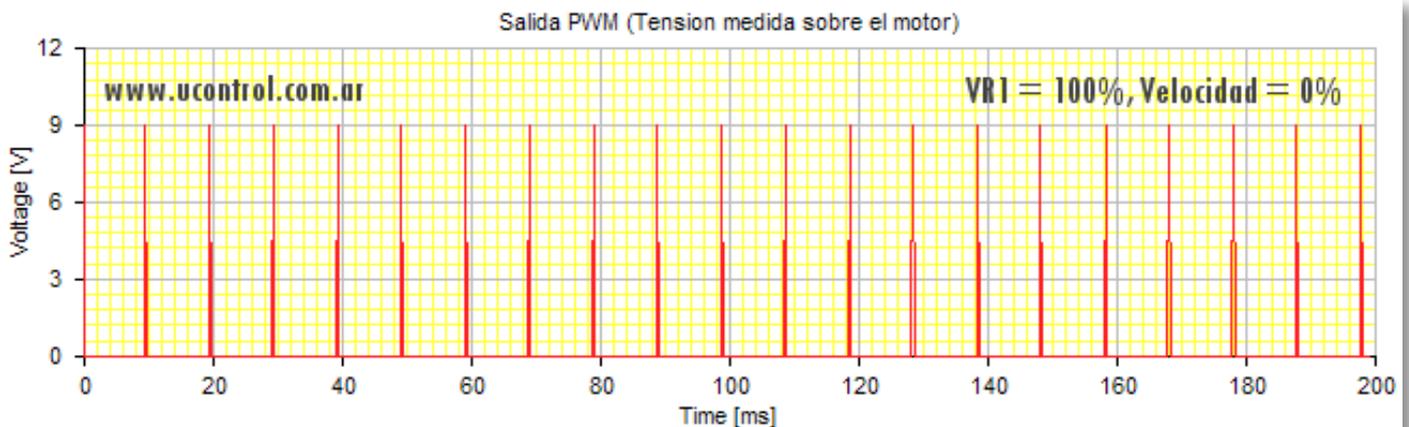



Figura 3: Forma de onda de la señal de salida del CI NE555, cuando R2 tiene el valor máximo.

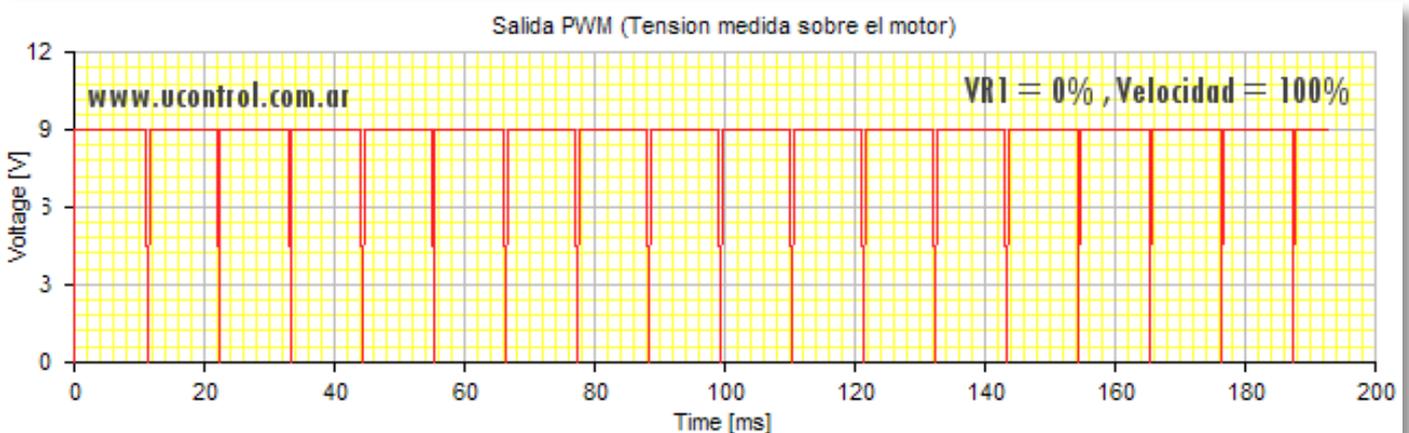



Figura 4: Forma de onda de la señal de salida del CI NE555, cuando R2 tiene el valor mínimo.

.Montaje

Hemos dibujado un circuito impreso y también un esquema con la posición de los componentes sobre la placa. Puedes ver el PCB en la **Figura 5** y la posición de los componentes en la **Figura 6.**

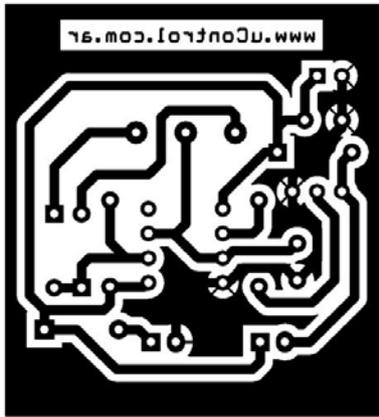



Figura 5: Este es el PCB a utilizar.

Figura 6: Puedes usar esta imagen para facilitar el montaje.

Para motores de mayor consumo de corriente, utilice el PCB mostrado en la **Figura 7** y el esquema de montaje de la **Figura 8**.

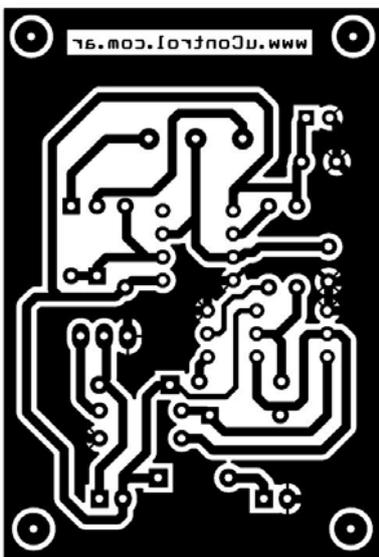



Figura 7: Este es el PCB a utilizar.

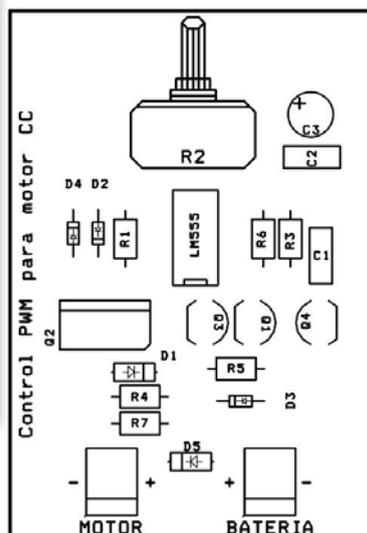



Figura 8: Puedes usar esta imagen para facilitar el montaje.

Lista de componentes necesarios para el montaje (fig. 5)

Cantidad	Referencia	Modelo
2	C1,C2	0.1uF
1	C3	10uF
3	D1,D2,D3	1N4148
1	D4	1N4004
1	J1	Bateria 6-12Vdc
1	MG1	MOTOR DC
1	Q1	2N2222
1	R1	1K
1	R2	100K
1	R3	22K
1	U1	LM555

Lista de componentes necesarios para el montaje (fig. 7)

Cantidad	Referencia	Modelo
2	C1,C2	0.1uF
1	C3	10uF
2	D1,D5	1N4004
3	D2,D3,D4	1N4148
1	J1	Batería 6-12Vdc
1	MG1	MOTOR DC
2	Q1,Q4	2N3906
1	Q2	IRFZ44
1	Q3	2N222
3	R1,R3,R7	1K
1	R2	100K
1	R4	100
1	R5	4.7K
1	R6	22K
1	U1	LM555